

UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
ESCOLA DE ENGENHARIA
DEPARTAMENTO DE ENGENHARIA ELÉTRICA

VICENTE COSTAMILAN DA CUNHA

**Métodos de Segmentação Automática de Sinais
de Eletromiografia de Superfície para
Classificação de Movimentos Utilizando Redes
Neurais Artificiais**

Porto Alegre

2015

UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
ESCOLA DE ENGENHARIA
DEPARTAMENTO DE ENGENHARIA ELÉTRICA

VICENTE COSTAMILAN DA CUNHA

**Métodos de Segmentação Automática de Sinais de
Eletromiografia de Superfície para Classificação de
Movimentos Utilizando Redes Neurais Artificiais**

Projeto de Diplomação apresentado ao Departamento de Engenharia Elétrica da Escola de Engenharia da Universidade Federal do Rio Grande do Sul, como requisito parcial para Graduação em Engenharia Elétrica

Orientador: Prof. Dr. Eng. Alexandre Balbinot

Porto Alegre

2015

UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
ESCOLA DE ENGENHARIA
DEPARTAMENTO DE ENGENHARIA ELÉTRICA

VICENTE COSTAMILAN DA CUNHA

Métodos de Segmentação Automática de Sinais de Eletromiografia de Superfície para Classificação de Movimentos Utilizando Redes Neurais Artificiais

Projeto de Diplomação apresentado ao Departamento de Engenharia Elétrica da Escola de Engenharia da Universidade Federal do Rio Grande do Sul, como requisito parcial para Graduação em Engenharia Elétrica

Trabalho aprovado. Porto Alegre, 03 de dezembro de 2015:

Prof. Dr. Eng. Alexandre Balbinot
Orientador

Prof. Dr. Eng. Altamiro Susin
Convidado

M.^a Eng. Gabriela Favieiro
Convidada

Eng. Vinicius Cene
Suplente

*A Gilberto, mi padre, torre de razón y de firme fe;
e a todos aqueles que tomarem interesse neste estudo.*

Agradecimentos

Aos demais colaboradores e pesquisadores do laboratório de Instrumentação Eletro-Eletrônica, em especial Vinicius Cene e Fernanda Trevisol, que desenvolveram a coleta da base de dados utilizada e prestaram auxílio de forma geral.

*Take nothing on its looks;
take everything on evidence.
There's no better rule.*

Charles Dickens, Great Expectations

Resumo

A segmentação de sinais de eletromiografia (EMG) é parte essencial da etapa de pré-processamento em aplicações de reconhecimento de movimentos e controle de próteses. Métodos de segmentação automática possibilitam a individualização de trechos de interesse do sinal correspondentes a esforços musculares e o descarte de trechos de sinal com baixa atividade muscular, por exemplo. Neste estudo, quatro métodos para segmentação automática de sinais de EMG de superfície, adaptados de outros trabalhos na área, foram propostos e implementados em MATLAB R2015a. Os métodos foram aplicados nos sinais da base de dados do projeto NinaPro e nos sinais da base de dados adquiridos pelo Laboratório de Instrumentação Eletro-Eletrônica da UFRGS. Redes neurais artificiais (RNAs) foram utilizadas para classificar os sinais segmentados com os quatro métodos entre 17 movimentos de mão e punho. **TODO: RESULTADOS.**

Palavras-chave: Eletromiografia de Superfície. Segmentação de Sinais. Redes Neurais Artificiais.

Abstract

TODO: TRADUZIR RESUMO.

Keywords: Surface Eletromiography. Signal Segmentation. Artificial Neural Networks.

Lista de Figuras

Figura 1 – Soma de potenciais de ação das n fibras de uma unidade motora, formando uma MUAP $h(t)$	16
Figura 2 – MUAPTs de diferentes MUs somam-se para compor o sinal adquirido por um canal de EMG.	17
Figura 3 – Fluxograma representativo do MTD1.	19
Figura 4 – Fluxograma representativo do MTD2.	21
Figura 5 – Fluxograma representativo do MTD3.	25
Figura 6 – Fluxograma representativo do MTD4.	26
Figura 7 – Diagrama representativo do modelo de um neurônio artificial genérico.	27
Figura 8 – Exemplo de uma RNA em arquitetura <i>feedforward</i>	27
Figura 9 – Posicionamento de eletrodos de superfície no braço de um voluntário.	29
Figura 10 – Cenas do vídeo apresentado aos voluntários na aquisição do Laboratório IEE.	30
Figura 11 – Diagrama de blocos geral para os métodos de segmentação MTD1 - MTD4.	30
Figura 12 – Exemplo para retificação e normalização de trecho de sinal de EMG.	32
Figura 13 – Diagrama de blocos para processo de classificação de movimentos com RNAs.	34

Lista de Tabelas

Tabela 1	–	Parâmetros utilizados para definir o MTD1.	18
Tabela 2	–	Parâmetros utilizados para definir o MTD2.	20
Tabela 3	–	Parâmetros utilizados para definir o MTD3.	22
Tabela 4	–	Parâmetros utilizados para definir o MTD4.	22
Tabela 5	–	Lista de movimentos de interesse apresentados aos voluntários.	31
Tabela 6	–	Parâmetros ajustáveis para os métodos de segmentação.	32
Tabela 7	–	Exemplo de matriz de confusão.	36
Tabela 8	–	Combinações de parâmetros utilizados nos métodos de segmentação. . .	37
Tabela 9	–	Combinações de parâmetros selecionados para cada base de dados. . .	38

Lista de Abreviaturas e Siglas

BEP	<i>Beginning Extraction Point</i>
DBSCAN	<i>Density-Based Spatial Clustering of Applications with Noise</i>
EEP	<i>Ending Extraction Point</i>
EMG	Eletromiografia
EMGs	Eletromiografia de Superfície
IEE	Laboratório de Instrumentação Eletro-Eletrônica
MU	<i>Motor Unit</i>
MUAP	<i>Motor Unit Action Potencial</i>
MUAPT	<i>Motor Unit Action Potencial Trains</i>
MTD#	<i>Método Número #</i>
NinaPro	<i>Non-Invasive Adaptive Prosthetics project</i>
RNA	Rede Neural Artificial

Sumário

1	INTRODUÇÃO	14
2	REVISÃO BIBLIOGRÁFICA	16
2.1	Sinais de Eletromiografia	16
2.2	Métodos de Segmentação	17
2.2.1	Método 1 - método iterativo utilizando <i>thresholding</i> para detecção de centros de segmentos de comprimento constante (MTD1)	18
2.2.2	Método 2 - método não iterativo utilizando <i>thresholding</i> para detecção de centros de segmentos de comprimento constante (MTD2)	20
2.2.3	Método 3 - método com janela deslizante para detecção de BEP e EEP de segmentos utilizando variação total (MTD3)	20
2.2.4	Método 4 - método com janela deslizante para detecção de BEP e EEP de segmentos utilizando <i>thresholding</i> (MTD4)	22
2.3	Princípios Básicos sobre Redes Neurais Artificiais	23
3	METODOLOGIA EXPERIMENTAL	28
3.1	Bases de Dados Utilizadas	28
3.1.1	Posicionamento de Eletrodos	28
3.1.2	Sistema para Aquisição de EMGs no IEE	28
3.1.3	Movimentos de Interesse	28
3.2	Métodos de Segmentação	29
3.2.1	Preprocessamento	29
3.2.2	Parâmetros Ajustáveis	30
3.2.3	Agrupamento das Posições de Segmentos de Diferentes Canais Utilizando DBSCAN	33
3.3	Classificação de Segmentos com Redes Neurais Artificiais	34
3.3.1	Características Utilizadas como Preditores	34
3.3.2	Classes de Movimentos Utilizadas como Resposta	35
3.3.3	Separação de Grupos de Treino, Validação e Teste para Treinamento	35
3.3.4	Estrutura de RNAs utilizada	35
3.3.5	Sensitividade de RNAs	36
4	RESULTADOS E DISCUSSÕES	37
4.1	Número de Segmentos Obtidos por Movimento e Combinação de Parâmetros	37
4.2	Classificação de Movimentos Utilizando RNAs	38

5	CONCLUSÕES	39
6	PROPOSTAS DE TRABALHOS FUTUROS	40
	REFERÊNCIAS	41
	APÊNDICES	44
	APÊNDICE A – FUNÇÃO EM MATLAB PARA MTD1	45
	APÊNDICE B – FUNÇÃO EM MATLAB PARA MTD2	48
	APÊNDICE C – FUNÇÃO EM MATLAB PARA MTD3	51
	APÊNDICE D – FUNÇÃO EM MATLAB PARA MTD4	54
	APÊNDICE E – CÓDIGO UTILIZADO PARA MEDIDA DE NÚ- MERO DE SEGMENTOS OBTIDOS PARA DI- FERENTES PARÂMETROS COM MTD1	55
	APÊNDICE F – CÓDIGO UTILIZADO PARA MEDIDA DE NÚ- MERO DE SEGMENTOS OBTIDOS PARA DI- FERENTES PARÂMETROS COM MTD2	57
	APÊNDICE G – CÓDIGO UTILIZADO PARA MEDIDA DE NÚ- MERO DE SEGMENTOS OBTIDOS PARA DI- FERENTES PARÂMETROS COM MTD3	59
	APÊNDICE H – CÓDIGO UTILIZADO PARA MEDIDA DE NÚ- MERO DE SEGMENTOS OBTIDOS PARA DI- FERENTES PARÂMETROS COM MTD4	61
	APÊNDICE I – CÓDIGO UTILIZADO PARA OBTENÇÃO DE RE- SULTADOS DE CLASSIFICAÇÃO UTILIZANDO RNA COM MTD1	62
	APÊNDICE J – CÓDIGO UTILIZADO PARA OBTENÇÃO DE RE- SULTADOS DE CLASSIFICAÇÃO UTILIZANDO RNA COM MTD2	65
	APÊNDICE K – CÓDIGO UTILIZADO PARA OBTENÇÃO DE RE- SULTADOS DE CLASSIFICAÇÃO UTILIZANDO RNA COM MTD3	68

APÊNDICE L – CÓDIGO UTILIZADO PARA OBTENÇÃO DE RESULTADOS DE CLASSIFICAÇÃO UTILIZANDO RNA COM MTD4	71
APÊNDICE M – CÓDIGO UTILIZADO PARA DETERMINAÇÃO DE r_{target} DO MTD1	72
APÊNDICE N – FUNÇÃO EM MATLAB PARA AGRUPAMENTO COM DBSCAN	73
APÊNDICE O – FUNÇÃO EM MATLAB PARA OBTENÇÃO DOS MOVIMENTOS RELACIONADOS AOS SEGMENTOS	76

1 Introdução

Sinais de EMG apresentaram crescentes aplicações no controle de próteses mioelétricas. (HARGROVE et al., 2013) mostraram o controle de uma prótese de perna de um amputado acima do joelho direito, enquanto (CHU et al., 2007) apresentaram bons resultados de reconhecimento de padrões de EMG para desenvolvimento de uma prótese multifuncional de mão. Em área paralela ao controle de próteses, (PATTICHIS; SCHIZAS; MIDDLETON, 1995) realizam diagnósticos clínicos de desordens neuromusculares com sinais de EMG e redes neurais artificiais.

As principais estratégias para caracterização de sinais de EMG e potenciais de ação das unidades motoras baseiam-se no uso de um método classificador. Métodos de classificação utilizados incluem - entre inúmeros outros - Redes Neurais Artificiais (HUDGINS; PARKER; SCOTT, 1993), classificador Bayesiano (ENGLEHART; HUDGINS, 2003), lógica *Fuzzy* (CHAN et al., 2000) e *Neuro-Fuzzy* (FAVIEIRO; BALBINOT, 2011). Tais sistemas de classificação necessitam, como parte do préprocessamento, segmentar os sinais de EMG adquiridos, para então realizar extração de características dos segmentos como amplitude, número de cruzamentos por zero, coeficientes de autoregressão, transformadas de Fourier e, mais recentemente, transformadas Wavelet (CHU et al., 2007).

Neste trabalho, é proposto e implementado em MATLAB quatro diferentes métodos de segmentação automática para sinais de EMG de superfície (EMGs). Os primeiros dois métodos (que serão identificados neste estudo pelas abreviações MTD1 e MTD2) tratam da detecção de picos do sinal utilizando *thresholding* e produzem segmentos de comprimento constante centrados nestes picos. O terceiro (MTD3) e quarto (MTD4) métodos utilizam uma janela deslizante para identificação de pontos iniciais e finais dos segmentos, produzindo segmentos de comprimento variável.

O primeiro método (MTD1) é baseado no método de segmentação utilizado em (CHAUVET et al., 2001). Trata-se de método iterativo, identificando os picos do sinal a partir de *threshold* de amplitude, segmentando o sinal em janelas de comprimento constante centradas nos picos. O valor de *threshold* para a primeira iteração corresponde ao máximo absoluto do sinal. A cada nova iteração em que não se atinge uma razão mínima arbitrada entre número de segmentos e comprimento de sinal, o novo *threshold* é calculado como fração do *threshold* da iteração anterior.

O segundo método (MTD2) é baseado no método de segmentação utilizado em (KATSIS et al., 2006). De forma similar ao MTD1, também utiliza *threshold* para detecção de picos do sinal e segmentação com janelas de comprimento constante em torno dos picos.

Diferentemente do MTD1, MTD2 não é iterativo, utilizando relação entre valor máximo e valor médio do sinal para cálculo do valor de *threshold*.

O terceiro método (MTD3) é baseado no método de segmentação utilizado em (GUT; MOSCHYTZ, 2000). Uma janela deslizante percorre o sinal, identificando inícios de segmentos quando a variação total no interior da janela excede determinado valor limite. Os finais dos segmentos são identificados quando a variação total do sinal no interior da janela é inferior a um segundo valor de limite.

O quarto método (MTD4) é baseado no método de segmentação utilizado em (PATTICHIS; SCHIZAS; MIDDLETON, 1995). Os pontos de início do segmento são tais que, em uma janela à esquerda do ponto, o sinal mantém-se abaixo de determinado *threshold*. Os pontos de final de segmento, de forma similar, são tais que, em uma janela à direita do ponto, o sinal mantém-se abaixo do *threshold*.

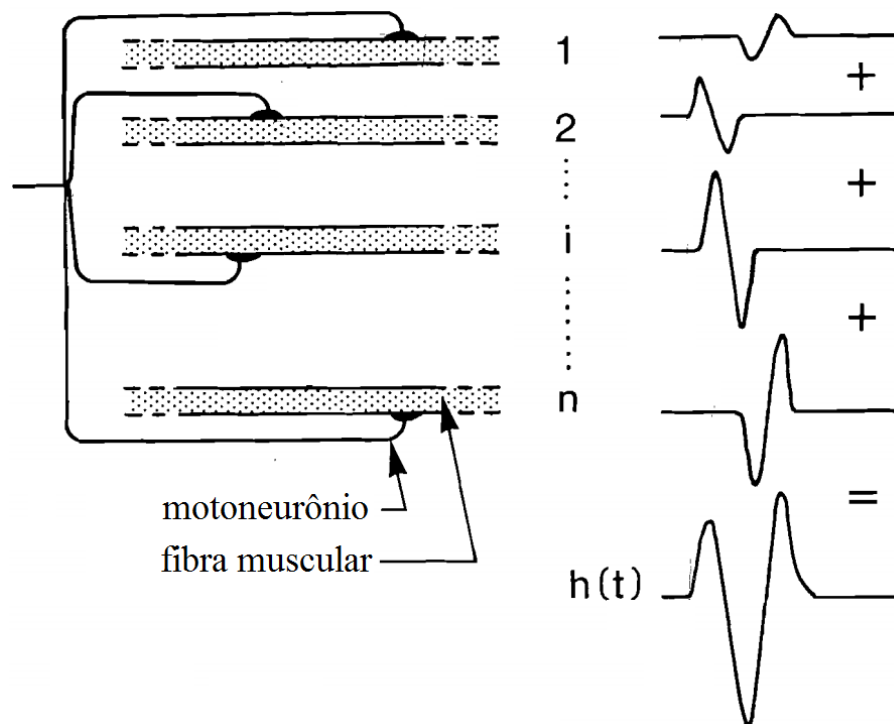
Utilizando valores de RMS, variância e frequência mediana dos segmentos obtidos, Redes Neurais Artificiais (RNAs) foram treinadas para classificar entre 17 movimentos de mão e punho. Os objetivos finais deste trabalho são a implementação dos métodos de segmentação propostos e fornecer avaliação comparativa entre métodos quando utilizados para classificação com uso de RNA.

2 Revisão Bibliográfica

2.1 Sinais de Eletromiografia

Sinais de EMG podem ser adquiridos por eletrodos posicionados na superfície da pele (eletrodo não invasivo) ou por agulhas introduzidas no tecido muscular (eletrodo invasivo). Os sinais são formados por potenciais de ação de fibras musculares organizadas em unidades funcionais chamadas de “unidades motoras” (MU - *Motor Unit*) (LUCA et al., 2006). Uma MU é composta por um neurônio motor e as fibras musculares que ele inerva, sendo a entidade fundamental que controla a ativação de músculos estriados (BUCHTHAL; SCHMALBRUCH, 1980). A soma algébrica dos potenciais de ação de todas as fibras de uma unidade motora é chamada de “potencial de ação da unidade motora”, ou em inglês, MUAP (*Motor Unit Action Potential*) (ALMEIDA; BARRETO, 1997). A Figura 1 apresenta a composição de uma MUAP a partir da soma dos potenciais das fibras de uma unidade motora.

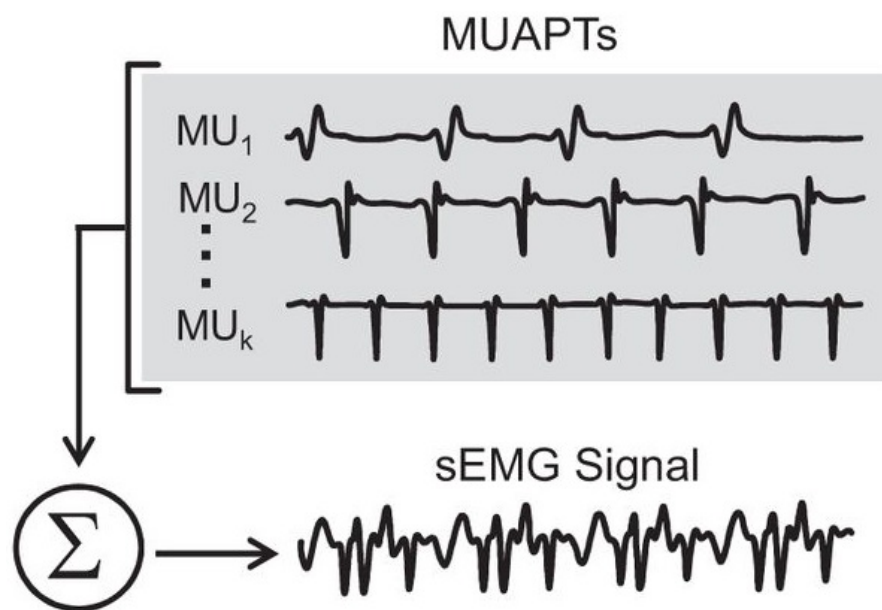
Figura 1 – Soma de potenciais de ação das n fibras de uma unidade motora, formando uma MUAP $h(t)$.



Fonte: adaptado de (BASMAJIAN; LUCA, 1985).

Para os principais métodos utilizados em aquisição do sinal de EMG, é comum a captura da contribuição de mais de uma unidade motora no mesmo canal de aquisição. A influência de uma unidade motora na amplitude do sinal adquirido depende principalmente da distância das fibras musculares ao ponto de aquisição (GERDLE et al., 1999). Sinais de EMG de longa duração são constituídos por sequências temporais de MUAPs, também conhecidas como MUAPTs (*MUAP Trains*). A Figura 2 exemplifica MUAPTs de diferentes MUs que somam-se para formar um sinal de EMG de longa duração.

Figura 2 – MUAPTs de diferentes MUs somam-se para compor o sinal adquirido por um canal de EMG.



Fonte: adaptado de (KLINE; LUCA, 2014)

Para maiores detalhes sobre sinais de eletromiografia de superfície, conceitos sobre anatomia e fisiologia sugere-se a consulta de referências clássicas na área e adicionalmente as seguintes referências (FAVIEIRO, 2009), (LOPES, 2014) e (SCHONS, 2014).

2.2 Métodos de Segmentação

Esta seção descreve os métodos de segmentação desenvolvidos, citando os trabalhos da área que foram utilizados como base teórica para os métodos.

Nota-se que nomes utilizados para variáveis e constantes (por exemplo, sinal a ser segmentado ' x ', *threshold* ' T ', etc.) foram determinados pelo autor deste trabalho, não necessariamente sendo os mesmos utilizados nos trabalhos citados.

Para as definições dos Métodos 3 e 4 (MTD3 e MTD4) são utilizados os termos BEP (*Beginning Extraction Point*, ponto inicial de um segmento) e EEP (*Ending Extraction Point*, ponto final de um segmento), também utilizados em (PATTICHIS; SCHIZAS; MIDDLETON, 1995).

2.2.1 Método 1 - método iterativo utilizando *thresholding* para detecção de centros de segmentos de comprimento constante (MTD1)

Este método iterativo é adaptado do método de segmentação utilizado em (CHAUVET et al., 2001). As definições da Tabela 1 serão utilizados para descrever este método.

Tabela 1 – Parâmetros utilizados para definir o MTD1.

Nome	Descrição
x	Sinal a ser segmentado
L	Comprimento total do sinal a ser segmentado
l	Comprimento desejado para os segmentos
T_k	Valor de <i>threshold</i> para a iteração k
T_{lim}	Valor de limite inferior para o <i>threshold</i>
q	Fração de T_{k-1} para determinação de T_k
N_k	Número total de candidatos para centros de segmentos identificados na iteração k
r_k	Razão entre número de candidatos identificados na iteração k e o comprimento total do sinal
r_{target}	Razão mínima esperada para r_k , utilizada para determinar o final do método

Inicialmente, determina-se o valor de *threshold* T_0 equivalente ao máximo absoluto do sinal a ser segmentado x (Equação (2.1)). O valor T_k é atualizado em cada iteração k como sendo uma fração q de T_{k-1} (Equação (2.2)). No trabalho de (CHAUVET et al., 2001), este valor q foi empiricamente determinado em 90%.

$$T_0 = \max(x) \quad (2.1)$$

$$T_k = q \times T_{k-1} \quad (2.2)$$

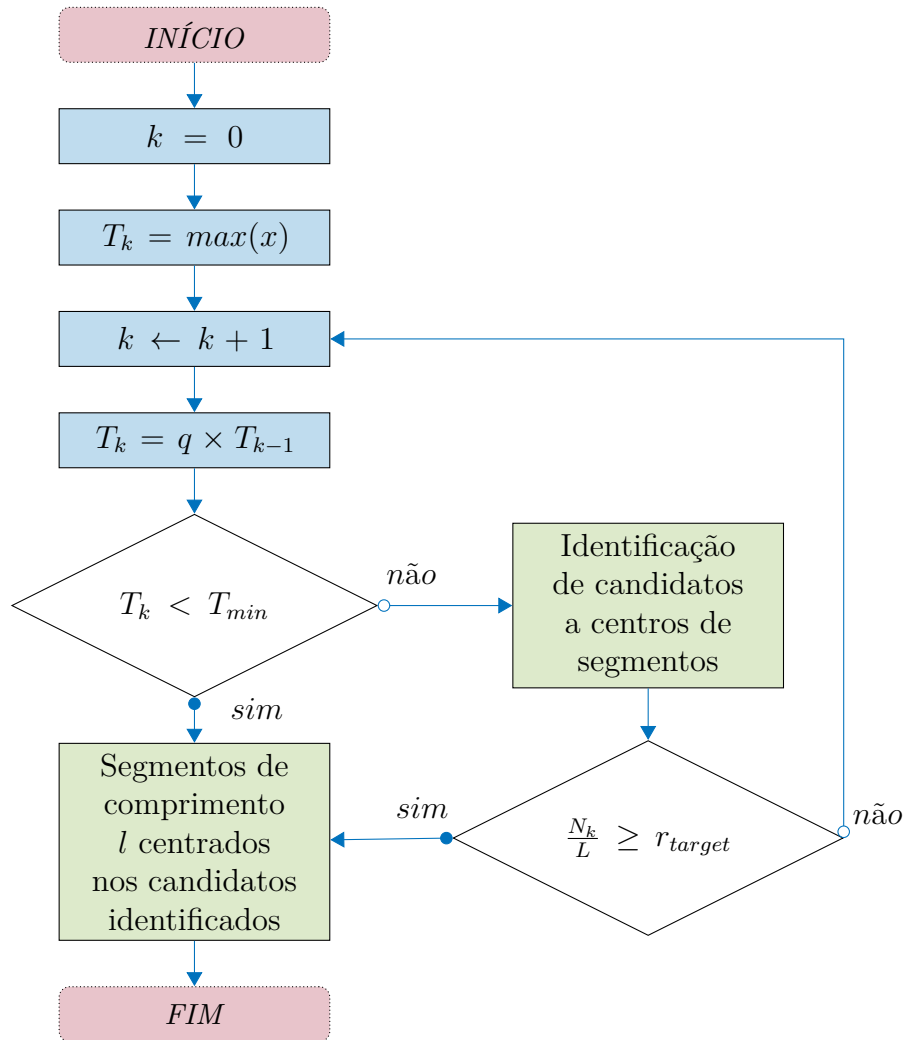
Pontos do sinal acima do valor de T_k são possíveis candidatos para centros de segmentos. Caso exista mais de um possível candidato em uma vizinhança bilateral de l amostras do sinal, apenas o ponto de maior amplitude nesta vizinhança é considerado. Para determinar o processo de finalização do método, avalia-se a razão r_k entre a quantidade identificada de candidatos N_k e o comprimento total do sinal L (Equação (2.3)). Caso r_k seja menor que um valor predeterminado r_{target} , calcula-se T_{k+1} para realização da próxima iteração (Equação (2.2)). Caso r_k seja maior ou igual ao valor predeterminado r_{target} ,

encerra-se o método e os segmentos são tomados como janelas de sinal de comprimento l , centradas nos candidatos identificados na última iteração.

$$r_k = \frac{N_k}{L} \quad (2.3)$$

Adicionalmente, o estabelecimento de um valor limite mínimo para *threshold* T_{lim} garante que o método não entre em laço infinito e evita detecção de segmentos em trechos de baixa atividade muscular. Caso o valor de *threshold* T_k para a iteração atual seja inferior a T_{lim} , encerra-se o processo iterativo. O método de segmentação MTD1 é representado pelo fluxograma da Figura 3.

Figura 3 – Fluxograma representativo do MTD1.



2.2.2 Método 2 - método não iterativo utilizando *thresholding* para detecção de centros de segmentos de comprimento constante (MTD2)

Este é o método de segmentação utilizado por (KATSIS et al., 2006), que será descrito pelas definições da Tabela 2. Primeiramente, seleciona-se entre dois métodos de cálculo de *threshold* T : ou utiliza-se T como múltiplo da média aritmética do sinal x ; ou T é uma fração do valor máximo do sinal x . (KATSIS et al., 2006) utilizaram a relação do fluxograma da Figura 4 para o cálculo de *threshold* T .

Tabela 2 – Parâmetros utilizados para definir o MTD2.

Nome	Descrição
x	Sinal a ser segmentado
L	Comprimento total do sinal a ser segmentado
l	Comprimento desejado para os segmentos
T	Valor de <i>threshold</i>
A	Coefficiente utilizado para decisão de método de cálculo de T
B	Múltiplo da média aritmética do sinal x para obtenção de T
C	Fração do valor máximo do sinal x para cálculo de T

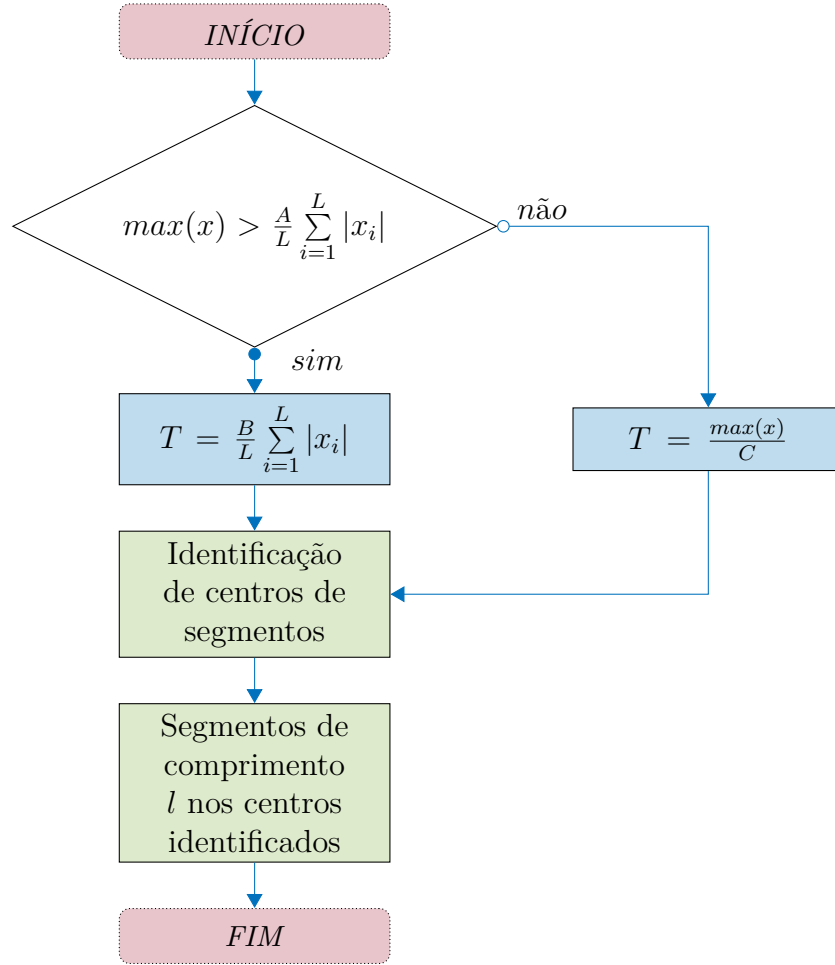
De forma similar ao MTD1, os pontos do sinal que tiverem valor acima de T são considerados possíveis candidatos para centros de segmentos. Para os possíveis candidatos que estiverem afastados de uma distância inferior a l , apenas o candidato de maior amplitude é considerado. Em (KATSIS et al., 2006) foram utilizados coeficientes A , B e C respectivamente de 30, 5 e 5, com comprimento l de 121 amostras.

2.2.3 Método 3 - método com janela deslizante para detecção de BEP e EEP de segmentos utilizando variação total (MTD3)

Este é o método de segmentação utilizado em (GUT; MOSCHYTZ, 2000). As definições da Tabela 3 serão utilizados para descrever este método. Uma janela deslizante de comprimento W percorre o sinal da esquerda para a direita. A cada incremento de *step* amostras, caso a variação total V (Equação (2.4)) do trecho de sinal contido pela janela exceda um limite B (sendo $B > 0$), o ponto mais à esquerda da janela w_0 determina a BEP de um segmento. O EEP do correspondente segmento é então obtido como o ponto mais à direita ($w_0 + W$) da próxima janela na qual a variação total for menor que um limite C (sendo $C < 0$). O MTD3 pode ser representado pelo fluxograma da Figura 5.

$$V = \sum_{i=w_0+1}^{w_0+W} (x_i - x_{i-1}) \quad (2.4)$$

Figura 4 – Fluxograma representativo do MTD2.



No método original de (GUT; MOSCHYTZ, 2000), BEPs foram detectadas pelo cálculo da declividade média β no interior da janela (Equação (2.5)) nos pontos que excediam um limite B (sendo $B > 0$), e EEPs pelo cálculo do módulo da variação total (Equação (2.6)) quando esta era menor que um limite C (sendo $C > 0$).

$$\beta = \frac{1}{W} \sum_{i=w_0+1}^{w_0+W} (x_i - x_{i-1}) \quad (2.5)$$

$$\gamma = \left| \sum_{i=w_0+1}^{w_0+W} (x_i - x_{i-1}) \right| \quad (2.6)$$

Nota-se que $\gamma = |W\beta| = |V|$. O MTD3 proposto, ao utilizar apenas a Equação (2.4) para detecção de BEPs e EEPs, explora tal relação para simplificar o método original, com a vantagem adicional de que os limites B e C passam a ser da mesma ordem de grandeza (com o método de (GUT; MOSCHYTZ, 2000), B seria aproximadamente W vezes menor que C).

Tabela 3 – Parâmetros utilizados para definir o MTD3.

Nome	Descrição
x	Sinal a ser segmentado
l_{min}	Distância mínima entre BEPs e EEPs de um mesmo segmento
l_{max}	Distância máxima entre BEPs e EEPs de um mesmo segmento
W	Comprimento da janela deslizante utilizada pelo método
w_0	Número da amostra mais a esquerda da janela. Determina a posição instantânea da janela
$step$	Número de amostras para incrementar w_0 antes de novo cálculo de variação total
V	Variação total do sinal x contido na janela deslizante
β	Declividade média do sinal x contido na janela deslizante
B	Valor limite para declividade média que determina um BEP
γ	Módulo da variação total do sinal x contido na janela deslizante
C	Valor limite para variação total que determina um EEP

O incremento de $step$ amostras (ao invés do avanço de w_0 de uma em uma amostra) serve para reduzir o número de vezes em que é necessário cálculo de variação total V , simplificando o processamento. Os limites de l_{min} e l_{max} são necessários para evitar identificação incorreta da EEP relacionada a uma BEP (i.e. l_{max} evita que o segmento BEP-EEP contenha sinal respectivo a mais de um movimento e l_{min} que o sinal seja segmentado em meio a um movimento).

2.2.4 Método 4 - método com janela deslizante para detecção de BEP e EEP de segmentos utilizando *thresholding* (MTD4)

Este é o método de segmentação adaptado de (PATTICHIS; SCHIZAS; MIDDLETON, 1995). As definições da Tabela 4 serão utilizados para descrever este método.

Tabela 4 – Parâmetros utilizados para definir o MTD4.

Nome	Descrição
x	Sinal a ser segmentado
L	Comprimento total do sinal x
W	Comprimento da janela deslizante utilizada pelo método
T	Valor de <i>threshold</i>

Uma janela deslizante de comprimento W com início em w_0 percorre o sinal da esquerda para a direita. Os BEPs dos segmentos são pontos w_0 tais que o valor máximo do sinal contido pela janela permanece abaixo do valor de *threshold* T . Para sequências de pontos consecutivos que atendam esta especificação, seleciona-se o último ponto (ponto mais à direita). As EEPs são identificadas de forma similar, sendo os primeiros pontos

$w_0 + W$ após a identificação de uma BEP nos quais o sinal contido pela janela permanece abaixo do valor de *threshold* T . O fluxograma da Figura 6 representa o MTD4.

TODO: atualizar esta descrição com a versão final do método

2.3 Princípios Básicos sobre Redes Neurais Artificiais

Rede Neural Artificial (RNA) trata-se de um método computacional que, inspirado nos modelos de redes neurais biológicas, “aprende” a resolver determinados problemas (YEGNANARAYANA, 2009), apresentando aplicações em diversos tipos de sinais (e.g. sinais de áudio, imagem e, como é o caso deste trabalho, eletromiografia).

Propostas de utilização de RNA com sinais de EMG para classificação de movimentos datam antes de 1990, a exemplo do trabalho de (HIRAIWA; SHIMOHARA; TOKUNAGA, 1989), que menciona a vantagem de RNA sobre outros mecanismos de aprendizado computacional na classificação de padrões que não apresentam separabilidade linear. Desde então, múltiplos trabalhos na área já utilizaram RNA para classificação de movimentos utilizando EMG (e.g. (HUDGINS; PARKER; SCOTT, 1993), (SUBASI; YILMAZ; OZCALIK, 2006), (BU; FUKUDA, 2003)), apresentando resultados satisfatórios para taxa de acerto de classificação.

Uma RNA é modelada pelas comunicações de um conjunto de “neurônios” artificiais. Cada neurônio artificial, como mostrado na Figura 7, trata-se de uma função matemática que soma de forma ponderada n entradas com pesos w_n e um valor de *bias* B e utiliza o resultado desta soma em uma função dita “função de ativação” ((TANIKIĆ, 2012)).

As RNAs utilizadas neste trabalho apresentam arquitetura chamada *feedforward*, onde neurônios são estruturados em camadas consecutivas, de forma que neurônios de uma mesma camada recebem as saídas de neurônios da camada anterior. A Figura 8 exemplifica a estrutura de uma rede neural *feedforward* que utiliza m entradas para produzir n saídas, utilizando duas camadas de neurônios chamadas de “camadas ocultas” (i.e. a saída da camada é utilizada como entrada da próxima camada) e uma camada de saída.

(HORNIK, 1991) mostra que ao utilizar neurônios com função de ativação contínua, limitada e não constante, a taxa de acerto de classificação para uma RNA de arquitetura *feedforward* depende somente do número de neurônios artificiais utilizados nas camadas ocultas. Em aplicações de reconhecimento de padrões, normalmente utiliza-se uma função de ativação do tipo sigmoide (i.e. função cuja curva lembra a forma de um “S”). A função de ativação utilizada para RNAs, neste trabalho, é a função sigmoide logística *logsig()*, dada pela Equação (2.7).

$$\text{logsig}(x) = \frac{1}{1 + e^{-x}} \quad (2.7)$$

Uma RNA pode ser utilizada para resolver determinado problema após um processo de treinamento supervisionado. Conjuntos de dados de treinamento (entradas e respostas esperadas) são apresentados à RNA. Um algoritmo iterativo modifica os pesos e *bias* dos neurônios buscando combinação que otimize o desempenho da RNA para os dados de treinamento. Às iterações do algoritmo de treinamento, em que os dados de treinamento são reapresentados à RNA, dá-se o nome de *epoch*. O algoritmo utilizado para treinamento de RNAs, neste trabalho, chama-se *scaled conjugate gradient backpropagation* (MØLLER, 1993) (identificado em MATLAB como *trainscg*), que apresenta boa performance em problemas de reconhecimento de padrões.

Quando uma RNA é aplicada para classificação de séries temporais de sinais, as entradas (chamadas “preditores”) utilizadas pela RNA são características extraídas de um trecho do sinal (e.g. número de cruzamentos por zero, valor médio, valor RMS). A extração de características tem o objetivo de reduzir a dimensionalidade do vetor de entradas da RNA (i.e. ao invés de um trecho de sinal com grande número de amostras, a entrada da RNA trata-se de um número arbitrariamente pequeno de preditores), o que apresenta vantagens em desempenho computacional e contorna problemas que surgem da classificação de sinais ruidosos (KIM; HAN, 2000).

Para maiores detalhes sobre RNAs e outros métodos computacionais, sugere-se consulta das seguintes referências. **TODO: referências de RNA.**

Figura 5 – Fluxograma representativo do MTD3.

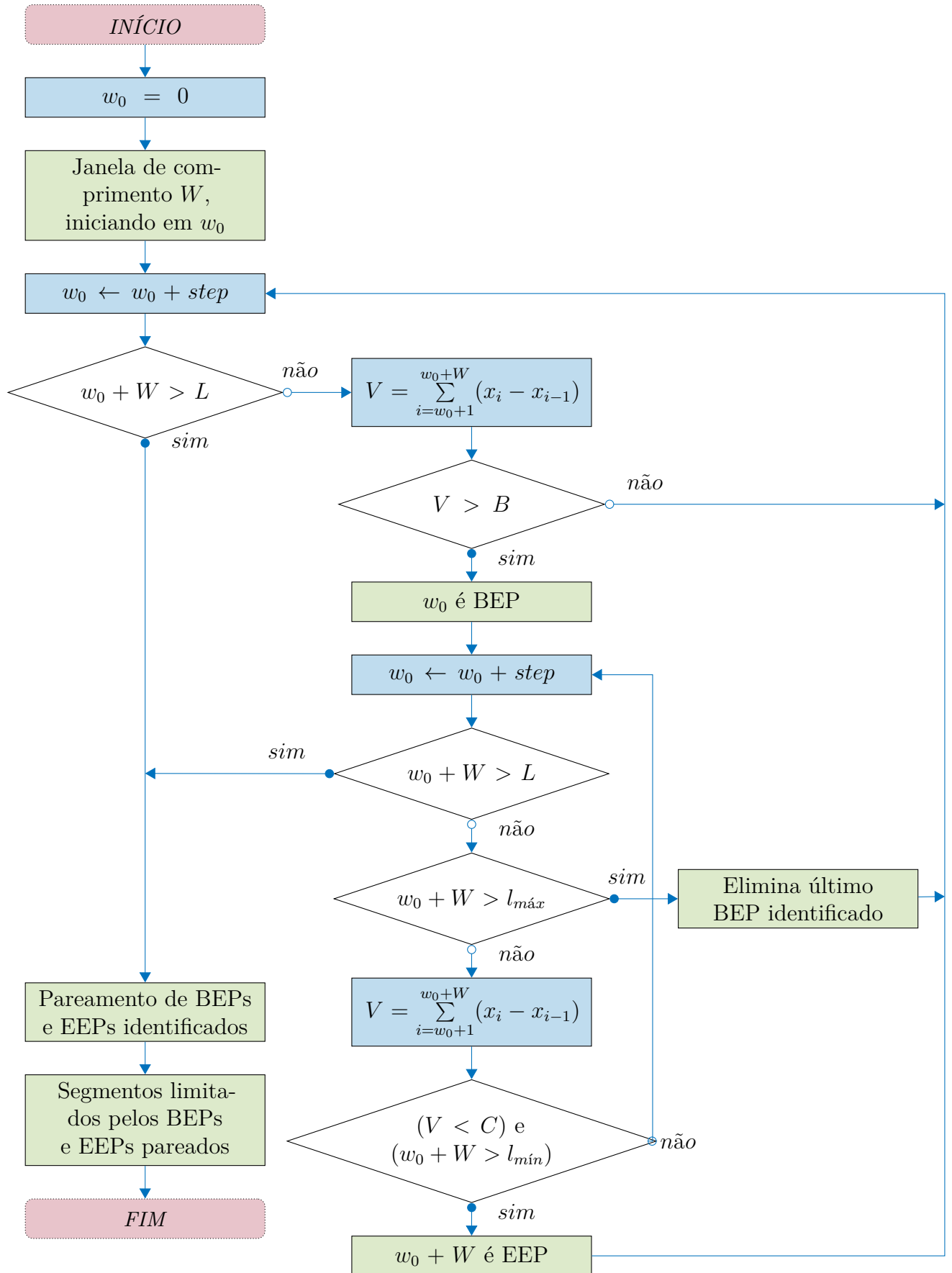


Figura 6 – Fluxograma representativo do MTD4.

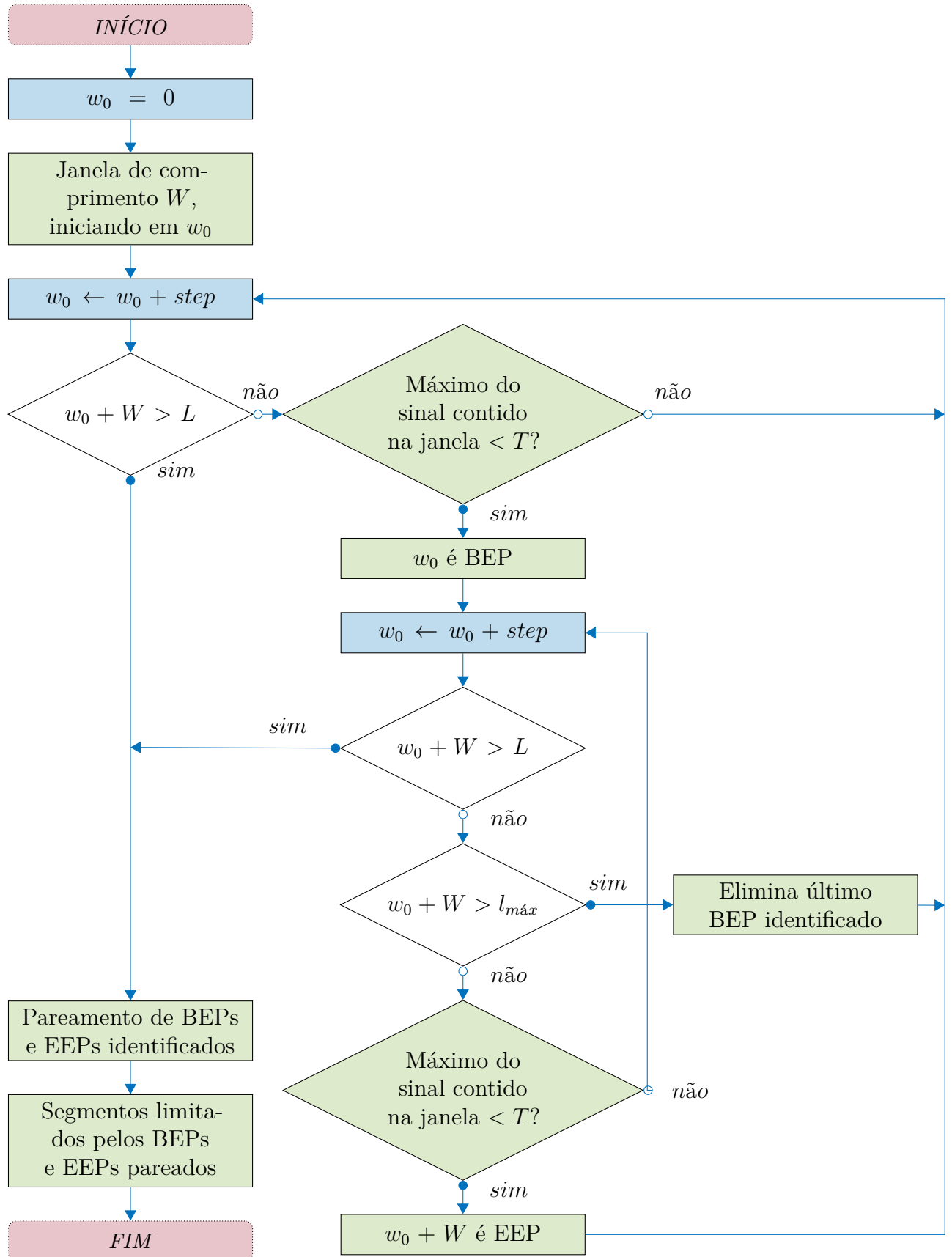
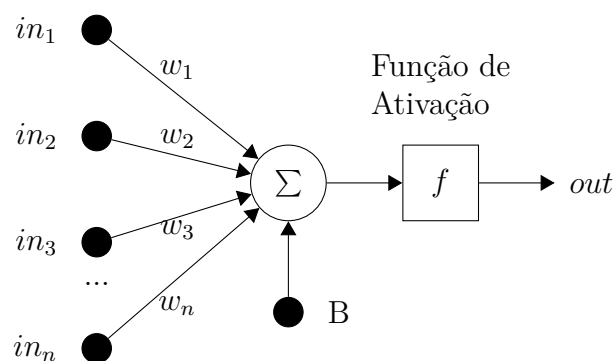
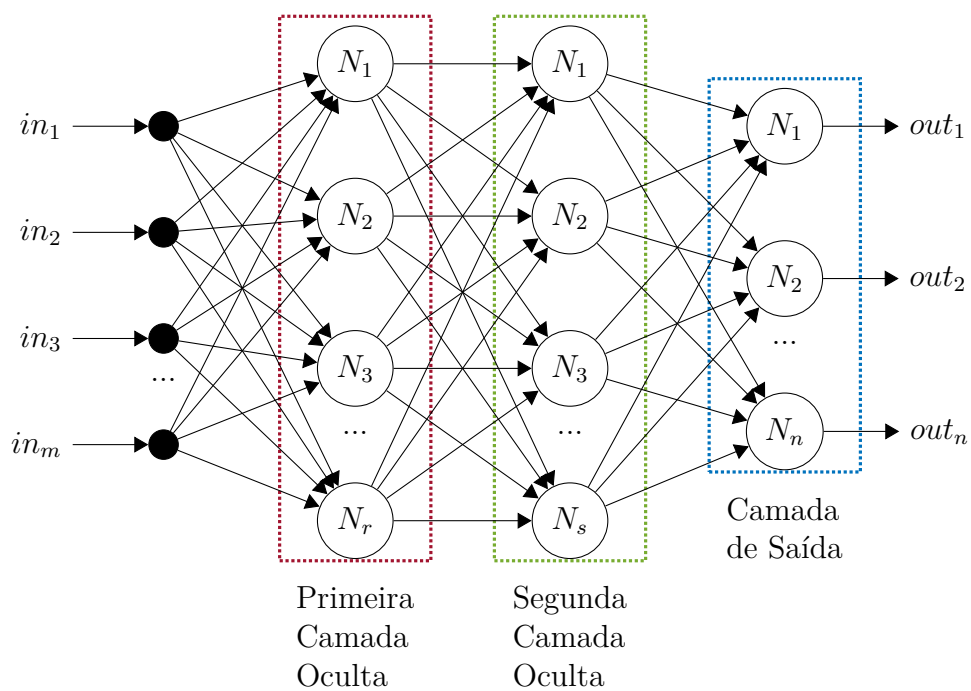


Figura 7 – Diagrama representativo do modelo de um neurônio artificial genérico.

Figura 8 – Exemplo de uma RNA em arquitetura *feedforward*.

3 Metodologia Experimental

3.1 Bases de Dados Utilizadas

Este trabalho faz uso de aquisições para o exercício 1 da base de dados número 2 do projeto NinaPro (GIJSBERTS et al., 2014) e a base de dados em construção realizada pelo Laboratório de Instrumentação Eletro-Eletrônica (IEE). A base de dados do IEE busca replicar os métodos de aquisição utilizados pelo projeto NinaPro, contando com os mesmos movimentos realizados, mesmo posicionamento de eletrodos e mesmo período de amostragem ($500 \mu s$).

3.1.1 Posicionamento de Eletrodos

Os sinais de EMG de superfície para ambas as bases utilizadas neste trabalho são compostos por canais de aquisição de 12 eletrodos posicionados no braço direito de voluntários saudáveis (i.e. não amputados e sem desordens neuromusculares). Os primeiros 8 canais correspondem a eletrodos posicionados de forma a circundar o antebraço e a junção úmero-radial. O eletrodo do canal número 9 é posicionado sobre o músculo flexor superficial dos dedos e o eletrodo do canal 10, em oposição, é posicionado sobre o músculo extensor dos dedos. Os últimos dois eletrodos, 11 e 12, são posicionados sobre o bíceps e o tríceps, respectivamente. A Figura 9 apresenta o posicionamento para os doze eletrodos de superfície e a numeração de seus respectivos canais que compõem os sinais para ambas as bases de dados.

3.1.2 Sistema para Aquisição de EMGs no IEE

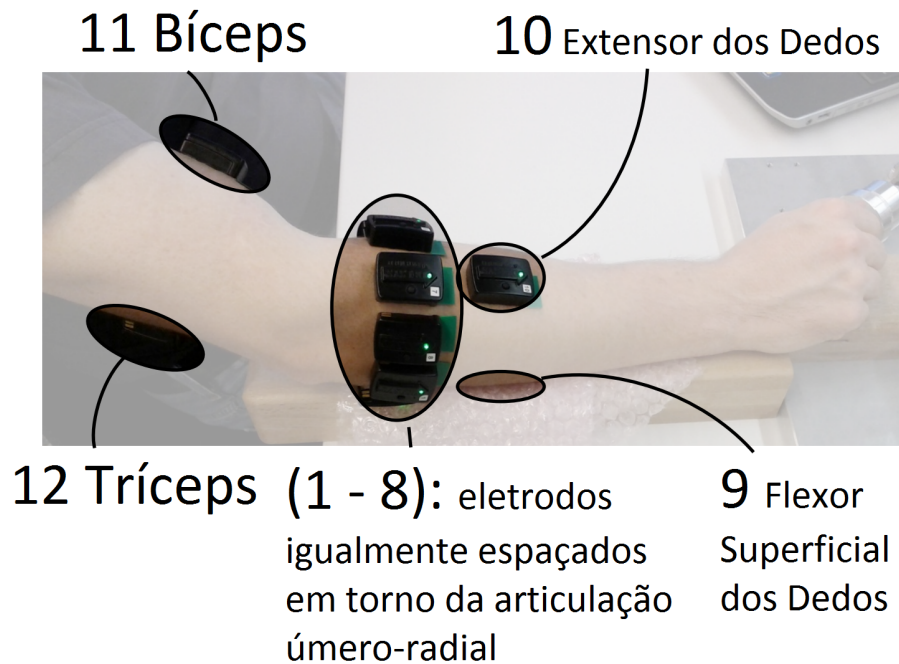
TODO: descrever o hardware utilizado na aquisição do IEE.

3.1.3 Movimentos de Interesse

Após o posicionamento de eletrodos, os voluntários sentam-se em frente a um monitor e apoiam o braço direito de forma relaxada sobre uma superfície horizontal próxima à altura do cotovelo. O monitor exibe um vídeo com uma mão direita, digitalmente animada, realizando movimentos de interesse que devem ser replicados pelos voluntários. A Figura 10 apresenta a mão digitalmente animada realizando alguns dos movimentos de interesse deste trabalho.

Primeiramente, realiza-se um “treinamento” com o voluntário, exibindo o vídeo e apresentando os movimentos que devem ser realizados (sem aquisição de sinal), de modo a

Figura 9 – Posicionamento de eletrodos de superfície no braço de um voluntário.



Fonte: adaptado de (GIJSBERTS et al., 2014)

reduzir possíveis erros na realização de movimentos para uma segunda exibição, quando os sinais de EMG serão devidamente adquiridos. Cada movimento apresentado em vídeo tem duração de 5 segundos, com um intervalo de 3 segundos de repouso entre movimentos. O voluntário realiza 6 repetições consecutivas para os 17 movimentos descritos na Tabela 5.

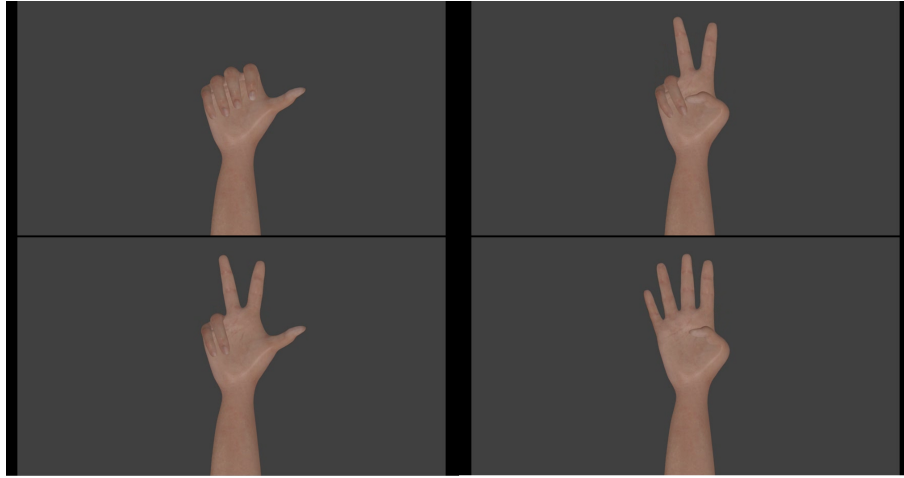
3.2 Métodos de Segmentação

Esta seção descreve a implementação dos métodos de segmentação propostos. O diagrama de blocos da Figura 11 apresenta de forma resumida os passos comuns aos quatro métodos, que serão explanados nas subseções seguintes. Os códigos criados para os quatro métodos, escritos para MATLAB R2015a, encontram-se nos Apêndices A a D.

3.2.1 Préprocessamento

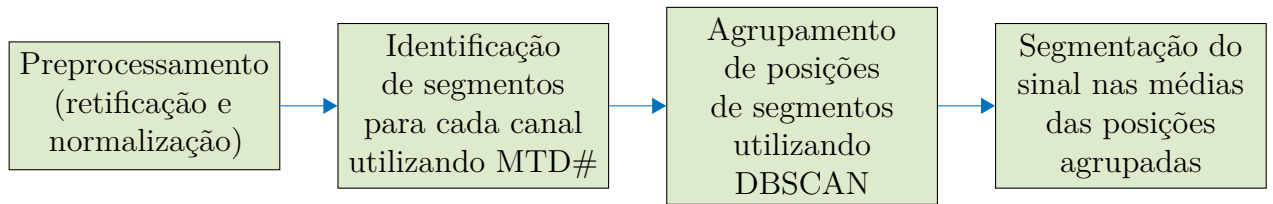
Os sinais de eletromiografia para ambas as bases de dados (NinaPro e IEE) são armazenados mantendo sua polaridade original (i.e. amostras do sinal podem assumir valores positivos e negativos). Primeiramente, realiza-se a retificação completa dos sinais tomando o módulo dos valores amostrados (em MATLAB, função *abs()*). A retificação completa do sinal mantém sua energia e é fundamental para a implementação dos métodos de segmentação aqui desenvolvidos.

Figura 10 – Cenas do vídeo apresentado aos voluntários na aquisição do Laboratório IEE.



Fonte: Laboratório de Instrumentação Eletro-Eletrônica, UFRGS

Figura 11 – Diagrama de blocos geral para os métodos de segmentação MTD1 - MTD4.


















Após a etapa de retificação, os sinais para cada canal de aquisição são normalizados de acordo com seu valor máximo, de modo que seu novo valor máximo seja unitário, a partir da Equação (3.1), onde x é o sinal original para um canal e x_{norm} é sua versão normalizada. A normalização de canais faz com que os parâmetros utilizados pelos métodos de segmentação sejam relativos ao valor máximo do sinal, possibilitando a implementação para diferentes voluntários. A Figura 12 exemplifica retificação e normalização para trecho de sinal de um canal de EMG.

$$x_{norm} = \frac{x}{\max(x)} \quad (3.1)$$

3.2.2 Parâmetros Ajustáveis

Cada método de segmentação MTD1 - MTD4 apresenta um conjunto de parâmetros ajustáveis. Tais parâmetros foram descritos anteriormente na Seção 2.2 (Tabelas 1 a 4). Após investigações iniciais das segmentações obtidas com diferentes valores de parâmetros, listou-se valores a serem explorados na aplicação de cada método, a fim de obter os

Tabela 5 – Lista de movimentos de interesse apresentados aos voluntários.

#	Descrição	Imagem demonstrativa	#	Descrição	Imagem demonstrativa	#	Descrição	Imagem demonstrativa
1	Polegar esticado, flexão dos outros dedos.		2	Extensão do indicador e dedo médio, flexão dos outros dedos.		3	Flexão do dedo anelar e mínimo, extensão dos outros dedos.	
4	Polegar para a base do dedo mínimo.		5	Abdução (“afastamento”) de todos os dedos estendidos.		6	Flexão de todos os dedos ao punho.	
7	Extensão do indicador em movimento de “apontar”.		8	Adução (“aproximação”) de todos os dedos estendidos.		9 10	Rotação de punho em torno do dedo médio, dois sentidos.	
11 12	Rotação de punho em torno do dedo mínimo, dois sentidos.		13	Flexão de punho.		14	Extensão de punho.	
15	Desvio radial do punho.		16	Desvio ulnar do punho.		17	Extensão de punho com mão cerrada.	

Fonte: adaptado de (ATZORI et al., 2014).

parâmetros mais adequados para cada uma das bases de dados, assim como avaliar quais classes de movimentos poderiam ser devidamente segmentados com cada método. A Tabela 6 apresenta os parâmetros ajustáveis de cada método e sua respectiva lista de valores explorados.

Para cada movimento de interesse, o voluntário realiza 6 repetições; portanto, espera-se que os métodos de segmentação produzam 6 segmentos por classe de movimento. Para testar as diferentes combinações de parâmetros, utilizou-se os códigos apresentados nos Apêndices E a H.

Figura 12 – Exemplo para retificação e normalização de trecho de sinal de EMG.

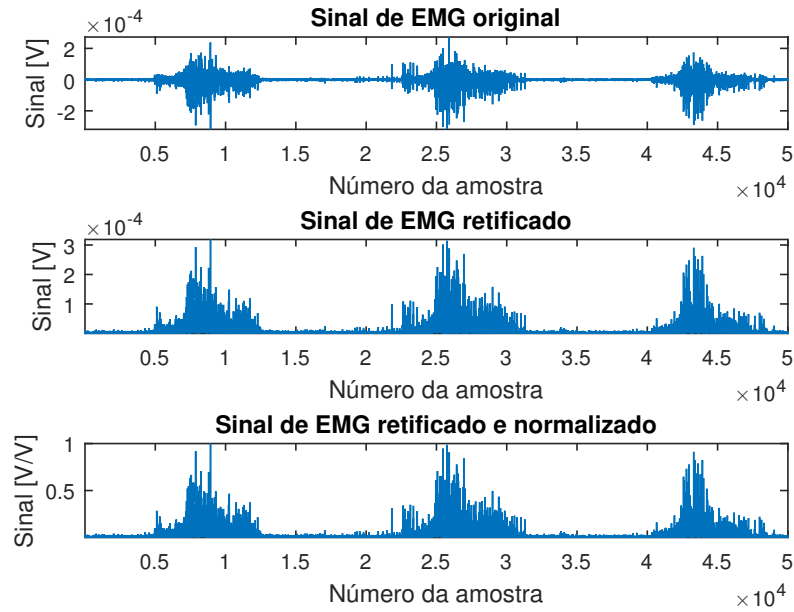


Tabela 6 – Parâmetros ajustáveis para os métodos de segmentação.

Método	Parâmetros	Valores utilizados	Número total de combinações
MTD1	l	10×10^3	16
	r_{target}	$5,6 \times 10^{-5}$	
	q	[0,8 0,85 0,9 0,95]	
	$T_{lim'}$	[0,05 0,1 0,15 0,2]	
MTD2	l	10×10^3	27
	A	[20 30 40]	
	B	[2 5 8]	
	C	[2 5 8]	
MTD3	W	5×10^3	16
	$step$	500	
	$l_{mín}$	$7,5 \times 10^3$	
	$l_{máx}$	$12,5 \times 10^3$	
	B	[0,1 0,2 0,3 0,4]	
MTD4	C	[-0,1 -0,2 -0,3 -0,4]	TODO
	TODO TODO	TODO TODO	

Para os métodos que necessitam parâmetro de comprimento de segmento l (i.e. os métodos que produzem segmentos de comprimento de janela constante, MTD1 e MTD2) utilizou-se valor de l como 10×10^3 , que corresponde a 5 cinco segundos de aquisição para ambas as bases de dados (período de amostragem para ambas é de $500 \mu s$), sendo a mesma duração dos segmentos de vídeo replicados pelos voluntários. O valor de MTD1 para r_{target}

foi obtido utilizando o código do Apêndice M como valor mínimo da razão entre número de segmentos e comprimento de sinal para as base de dados da NinaPro e IEE.

Para os métodos que utilizam janela deslizante (MTD3 e MTD4), utilizou-se comprimento de janela W de 5000 amostras, de modo que o comprimento temporal da janela (2,5 segundos) equivale à metade da duração do segmento de vídeo, com incrementos *step* de 500 amostras, ou um décimo do comprimento da janela. Os parâmetros de l_{\min} e l_{\max} utilizados foram de 7500 e 12500 amostras, respectivamente.

3.2.3 Agrupamento das Posições de Segmentos de Diferentes Canais Utilizando DBSCAN

Os sinais para ambas as bases de dados são compostos por 12 canais de aquisição. Os métodos de segmentação são implementados individualmente para os doze canais. Para os métodos MTD1 e MTD2, as posições centrais dos segmentos obtidas em cada canal são armazenadas, enquanto que para os métodos MTD3 e MT4 armazena-se as posições de BEPs e EEPs. Tais posições de diferentes canais são identificadas como pertencendo a um mesmo segmento através do algoritmo de grupamento DBSCAN (*Density-Based Spatial Clustering of Applications with Noise*).

DBSCAN foi inicialmente proposto em (ESTER et al., 1996) para grupamento de pontos utilizando dois parâmetros: ϵ , que define uma vizinhança em torno do valor de cada ponto, e *minPts*, que determina o número mínimo de pontos pertencentes à mesma vizinhança para formação do “núcleo” de um grupo. Para mais detalhes sobre DBSCAN, sugere-se o artigo original (ESTER et al., 1996) e a revisão do método (TRAN; DRAB; DASZYKOWSKI, 2013), cujo código em MATLAB foi utilizado (Apêndice N).

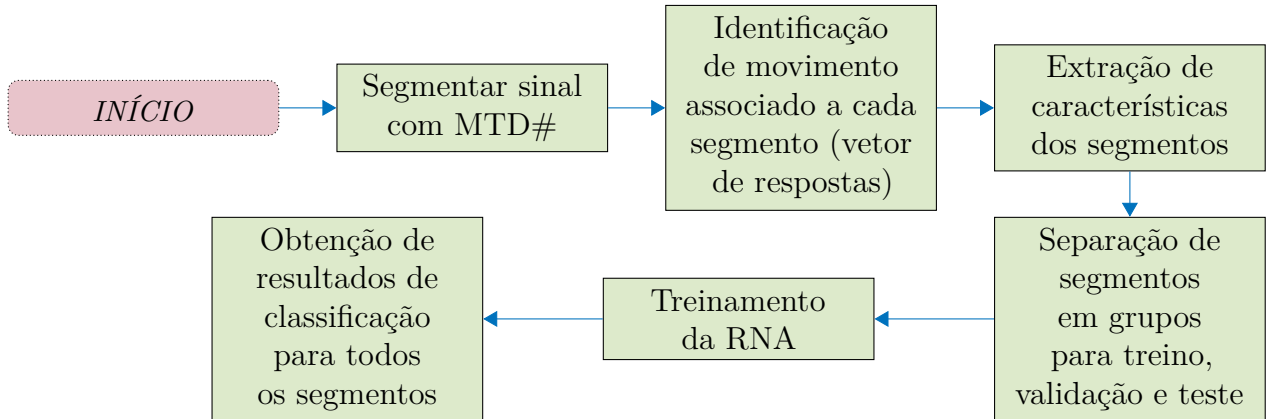
O algoritmo DBSCAN foi escolhido para esta aplicação por dois motivos principais. Primeiramente, ao contrário de métodos de grupamento como *k-means*, DBSCAN não necessita parâmetro que especifique número de grupos, no caso, número de segmentos a serem obtidos. Além disto, este método é capaz de descartar posições *outliers* que venham a ser identificadas para algum canal. Sendo assim, o algoritmo DBSCAN é capaz de tornar mais robusto os métodos de segmentação e implementáveis a sinais de múltiplos canais.

Após o agrupamento de posições de segmentos obtidos nos diferentes canais, tomam-se as médias de cada grupo como as posições nas quais o sinal deve ser segmentado (para os métodos MTD1 e MTD2, as médias dos grupos indicam posições centrais de segmentos e para métodos MTD3 e MTD4 indicam posições de BEPs e EEPs). O sinal é segmentado nestas posições para os doze canais, mantendo a coerência temporal entre canais.

3.3 Classificação de Segmentos com Redes Neurais Artificiais

Esta seção descreve o uso de RNAs para classificação dos segmentos obtidos de acordo com movimentos de interesse. O processo de classificação é representado pelo diagrama de blocos da Figura 13, que será explanado nas subseções seguintes.

Figura 13 – Diagrama de blocos para processo de classificação de movimentos com RNAs.



3.3.1 Características Utilizadas como Preditores

Os preditores (“entradas”) das RNAs são o valor RMS, a variância e a frequência mediana do espectro de potência dos segmentos de sinal obtidos pelos métodos MTD1 - MTD4. Tais características foram selecionados de acordo com outros trabalhos já realizados no Laboratório de Instrumentação Eletro-Eletrônica que obtiveram bons resultados em métodos de iteligência computacional, como (FAVIEIRO, 2009), (SCHONS, 2014) e (CENE, 2015).

O valor RMS ou *Root Mean Square* de um sinal discreto x de comprimento L (em MATLAB, função $rms()$) é dada pela Equação (3.2) e a variância deste sinal (em MATLAB, função $var()$) é dada pela Equação (3.3), onde \bar{x} é o valor médio do sinal. A frequência mediana é tal que a soma total da densidade de potência para frequências abaixo da frequência mediana é igual à soma total da densidade de potência para frequências acima da frequência mediana, sendo estimada pela função $medfreq()$ em MATLAB.

$$rms(x) = \sqrt{\frac{1}{L} \sum_{i=1}^L x_i^2} \quad (3.2)$$

$$var(x) = \frac{1}{L-1} \sum_{i=1}^L (x_i - \bar{x})^2 \quad (3.3)$$

3.3.2 Classes de Movimentos Utilizadas como Resposta

Realiza-se a identificação de movimentos associados a cada segmento obtido a partir de sua posição temporal no sinal original, já que durante a aquisição de dados os movimentos são realizados na mesma sequência para todos os voluntários. A base de dados NinaPro apresenta um vetor *stimulus* que indica os instantes em que foram apresentados cada movimento a ser replicado pelo voluntário. A partir deste vetor e das posições dos segmentos dentro do sinal original, utiliza-se o código apresentado no Apêndice O para criar o vetor de respostas esperadas no treinamento da RNA.

TODO: explicar o equivalente realizado para a base de dados IEE

3.3.3 Separação de Grupos de Treino, Validação e Teste para Treinamento

Os grupos de segmentos utilizadas no treinamento da rede neural são tais que:

- grupo de treino: primeiros quatro segmentos obtidos para uma mesma classe de movimento
- grupo de validação: quinto segmento obtido para cada classe de movimento
- grupo de teste: demais segmentos

Optou-se por este controle da divisão por indexação (método de divisão chamado em MATLAB de *divideind*) ao invés de métodos de divisão aleatória de grupos (*dividerand*) para evitar possíveis situações particulares nas quais todos (ou, na situação oposta e igualmente incômoda, nenhum) segmentos para determinada classe de movimento seria incluso no grupo “treino”, as quais viciariam resultados de classificação.

3.3.4 Estrutura de RNAs utilizada

As RNAs utilizadas para todos os métodos são do tipo *feedforward*, compostas por uma camada oculta e uma camada de saída, sendo a camada oculta formada por 40 neurônios com função de ativação *logsig()* (Equação (2.7)) e camada de saída por 17 neurônios, um para cada classe de movimento, de forma que o neurônio com saída de maior valor indica qual a classe de movimento em que foi classificado o segmento.

TODO: finalizar esta subseção; adicionar imagem do matlab mostran a arquitetura da RNA

3.3.5 Sensitividade de RNAs

Para avaliação de taxas de acerto de um sistema classificador, costumeiramente utiliza-se uma tabela conhecida como “matriz de confusão”. A Tabela 7 exemplifica uma matriz de confusão genérica.

Tabela 7 – Exemplo de matriz de confusão.

		Classificação obtida	
		Negativa	Positiva
Classificação esperada	Negativa	a	b
	Positiva	c	d

a : verdadeiros negativos

b : falsos positivos

c : falsos negativos

d : verdadeiros positivos

Fonte: adaptado de (KUBAT; HOLTE; MATWIN, 1998).

No caso das RNAs deste trabalho, que realizam classificação entre 17 classes de movimentos, a Tabela 7 pode ser obtida para cada uma das classes, onde “Positivo” indica a ocorrência da classe de movimento e “Negativo” a não-ocorrência da mesma. A taxa de acerto T_a para determinada classe de movimento é definida pela razão da soma de casos de verdadeiros positivos e verdadeiros negativos entre todos os casos classificados, como na equação (3.4).

$$T_a = \frac{a + d}{a + b + c + d} \quad (3.4)$$

Entretanto, utilizar (3.4) para quantificar a performance de classificação dos segmentos não é adequado para aplicações em que casos “negativos” e “positivos” não são igualmente distribuídos para uma série de dados de treino (e.g. se em 1000 casos espera-se 995 classificações negativas e 5 classificações positivas, se a RNA classificar todos os casos como negativos, errando classificação de todos os casos positivos, T_a terá valor 99,5%) (KUBAT; HOLTE; MATWIN, 1998).

Como cada série de treino corresponde à realização de 17 classes movimentos distintos em 6 repetições, o número de casos esperados negativos é, para cada classe, superior a dez vezes o número de casos esperados positivos. Para melhor quantificar a performance de classificação das RNAs, foi utilizada medida de sensibilidade r (equação (3.5)), definida como a fração de classificações verdadeiramente positivas entre todos os casos esperados positivos.

$$r = \frac{d}{c + d} \quad (3.5)$$

4 Resultados e Discussões

4.1 Número de Segmentos Obtidos por Movimento e Combinação de Parâmetros

Para as diferentes combinações de parâmetros listadas na Tabela 8, as Tabelas ?? a ?? **TODO: adicionar demais tabelas** apresentam o número médio de segmentos obtidos para todos os voluntários de cada base de dados utilizada.

Como os voluntários realizam 6 repetições de cada movimento, os melhores parâmetros são aqueles que melhor conseguirem identificar este número de segmentos por classe de movimento. Nas Tabelas ?? a ??, indica-se em verde as combinações de parâmetros e movimentos que foram apropriadamente segmentados para mais de 75% dos voluntários e em vermelho para menos de 25% dos voluntários.

Tabela 8 – Combinações de parâmetros utilizados nos métodos de segmentação.

Método#	1			2			3		4
Parâmetro	q	T_{lim}	A	B	C	B	C	T	
Índice da Combinação	1	0.80	0.05	20	2	2	0,01	-0,01	
	2	0.85	0.05	30	2	2	0,02	-0,01	
	3	0.90	0.05	40	2	2	0,03	-0,01	
	4	0.95	0.05	20	5	2	0,04	-0,01	
	5	0.80	0.10	30	5	2	0,01	-0,02	
	6	0.85	0.10	40	5	2	0,02	-0,02	
	7	0.90	0.10	20	8	2	0,03	-0,02	
	8	0.95	0.10	30	8	2	0,04	-0,02	
	9	0.80	0.15	40	8	2	0,01	-0,03	
	10	0.85	0.15	20	2	5	0,02	-0,03	
	11	0.90	0.15	30	2	5	0,03	-0,03	
	12	0.95	0.15	40	2	5	0,04	-0,03	
	13	0.80	0.20	20	5	5	0,01	-0,04	
	14	0.85	0.20	30	5	5	0,02	-0,04	
	15	0.90	0.20	40	5	5	0,03	-0,04	
	16	0.95	0.20	20	8	5	0,04	-0,04	
	17			30	8	5			
	18			40	8	5			
	19			20	2	8			
	20			30	2	8			
	21			40	2	8			
	22			20	5	8			
	23			30	5	8			
	24			40	5	8			
	25			20	8	8			
	26			30	8	8			
	27			40	8	8			

Na base de dados NinaPro, os métodos com segmentos de comprimento constante (MTD1 e MTD2) obtiveram boa segmentação do movimento 5 (abdução de todos os dedos) em diferentes combinações de parâmetros. No MTD2 não foi possível encontrar combinação de parâmetros apropriada para segmentação do movimento 11 (rotação de punho em torno do dedo mínimo). O MTD3 foi o método capaz de realizar segmentações apropriadas para um maior número de movimentos distintos. Percebe-se que os métodos MTD1 - MTD3 todos apresentaram dificuldades na segmentação dos movimentos 1 (polegar esticado) e 7 (extensão do indicador).

TODO: finalizar análise após obtenção de mais resultados

4.2 Classificação de Movimentos Utilizando RNAs

Para os resultados de segmentação obtidos com diferentes parâmetros testados apresentados nas Tabelas ?? a ??, identificou-se as combinações de parâmetros que apresentaram os resultados de segmentação mais satisfatórios (i.e. parâmetros que segmentaram em seis partes cada movimento para o maior número possível de voluntários da base) para serem usados nos códigos dos apêndices I a L, que realizam segmentação e classificação de movimentos utilizando RNA. As combinações de parâmetros selecionadas são apresentadas na Tabela 9.

Tabela 9 – Combinações de parâmetros selecionados para cada base de dados.

Método	Base de dados	Índice da combinação	Parâmetros
MTD1	NinaPro IEE	8 TODO	$q = 0.95$ $T_{lim} = 0.10$ TODO
MTD2	NinaPro IEE	8 TODO	$A = 30$ $B = 8$ $C = 2$ TODO
MTD3	NinaPro IEE	4 TODO	$B = 0.04$ $C = -0.01$ TODO
MTD4	NinaPro IEE	TODO TODO	TODO TODO

5 Conclusões

6 Propostas de Trabalhos Futuros

PROPOSTAS: tradução dos códigos para outras linguagens (e.g. C++, Python);
adaptação dos métodos para versões causais; aplicação no controle de prótese

Referências

- ALMEIDA, M.; BARRETO, J. *Filtragem digital de sinais biomédicos*. Universidade Federal de Santa Catarina, Centro Tecnológico., 1997. Disponível em: <<https://books.google.com.br/books?id=L5iqkgEACAAJ>>. Citado na página 16.
- ATZORI, M. et al. Electromyography data for non-invasive naturally-controlled robotic hand prostheses. *Scientific Data*, Macmillan Publishers Limited SN -, v. 1, p. 140053 EP -, Dec 2014. Data Descriptor. Disponível em: <<http://dx.doi.org/10.1038/sdata.2014.53>>. Citado na página 31.
- BASMAJIAN, J.; LUCA, C. D. *Muscles Alive*. 5. ed. Baltimore: Williams and Wilkins, 1985. Citado na página 16.
- BU, N.; FUKUDA, O. Emg-based motion discrimination using a novel recurrent neural network. *J. Intell. Inf. Syst*, 2003. Citado na página 23.
- BUCHTHAL, F.; SCHMALBRUCH, H. Motor unit of mammalian muscle. *Physiological Reviews*, American Physiological Society, v. 60, n. 1, p. 90–142, 1980. Citado na página 16.
- CENE, V. H. Upper-limb movement classification based on semg signal validation with continuous channel selection. In: LA-CCI. *Annals of 2nd Latin-American Congress on Computational Intelligence*. Curitiba, PR, 2015. ISBN: 978-85-69972-00-6. Citado na página 34.
- CHAN, F. H. et al. Fuzzy EMG classification for prosthesis control. *IEEE Trans Rehabil Eng*, v. 8, n. 3, p. 305–311, Sep 2000. Citado na página 14.
- CHAUVET, E. et al. A method of emg decomposition based on fuzzy logic. In: *Engineering in Medicine and Biology Society, 2001. Proceedings of the 23rd Annual International Conference of the IEEE*. [S.l.: s.n.], 2001. v. 2, p. 1948–1950 vol.2. ISSN 1094-687X. Citado 2 vezes nas páginas 14 e 18.
- CHU, J.-U. et al. A supervised feature-projection-based real-time emg pattern recognition for multifunction myoelectric hand control. *Mechatronics, IEEE/ASME Transactions on*, v. 12, n. 3, p. 282–290, June 2007. ISSN 1083-4435. Citado na página 14.
- ENGLEHART, K.; HUDGINS, B. A robust, real-time control scheme for multifunction myoelectric control. *Biomedical Engineering, IEEE Transactions on*, v. 50, n. 7, p. 848–854, July 2003. ISSN 0018-9294. Citado na página 14.
- ESTER, M. et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In: . [S.l.]: AAAI Press, 1996. p. 226–231. Citado na página 33.
- FAVIEIRO, G.; BALBINOT, A. Adaptive neuro-fuzzy logic analysis based on myoelectric signals for multifunction prosthesis control. In: *Engineering in Medicine and Biology Society, EMBC, 2011 Annual International Conference of the IEEE*. [S.l.: s.n.], 2011. p. 7888–7891. ISSN 1557-170X. Citado na página 14.

FAVIEIRO, G. W. *Controle de uma Prótese Experimental do Segmento Mão-Braço por Sinais Mioelétricos e Redes Neurais Artificiais*. 2009. Trabalho de Diplomação, Curso de Engenharia de Computação, UFRGS. Citado 2 vezes nas páginas 17 e 34.

GERDLE, B. et al. Acquisition, processing and analysis of the surface electromyogram. In: WINDHORST, U.; JOHANSSON, H. (Ed.). *Modern Techniques in Neuroscience Research*. Springer Berlin Heidelberg, 1999. p. 705–755. ISBN 978-3-642-63643-1. Disponível em: <http://dx.doi.org/10.1007/978-3-642-58552-4_26>. Citado na página 17.

GIJSBERTS, A. et al. Measuring movement classification performance with the movement error rate. *IEEE Transactions on neural systems and rehabilitation engineering*, 2014. Citado 2 vezes nas páginas 28 e 29.

GUT, R.; MOSCHYTZ, G. S. High-precision emg signal decomposition using communication techniques. *Signal Processing, IEEE Transactions on*, v. 48, n. 9, p. 2487–2494, Sep 2000. ISSN 1053-587X. Citado 3 vezes nas páginas 15, 20 e 21.

HARGROVE, L. J. et al. Robotic leg control with emg decoding in an amputee with nerve transfers. *New England Journal of Medicine*, v. 369, n. 13, p. 1237–1242, 2013. PMID: 24066744. Disponível em: <<http://dx.doi.org/10.1056/NEJMoa1300126>>. Citado na página 14.

HIRAIWA, A.; SHIMOHARA, K.; TOKUNAGA, Y. Emg pattern analysis and classification by neural network. In: *Systems, Man and Cybernetics, 1989. Conference Proceedings., IEEE International Conference on*. [S.l.: s.n.], 1989. p. 1113–1115 vol.3. Citado na página 23.

HORNIK, K. Approximation capabilities of multilayer feedforward networks. *Neural Networks*, v. 4, n. 2, p. 251 – 257, 1991. ISSN 0893-6080. Disponível em: <<http://www.sciencedirect.com/science/article/pii/089360809190009T>>. Citado na página 23.

HUDGINS, B.; PARKER, P.; SCOTT, R. A new strategy for multifunction myoelectric control. *Biomedical Engineering, IEEE Transactions on*, v. 40, n. 1, p. 82–94, Jan 1993. ISSN 0018-9294. Citado 2 vezes nas páginas 14 e 23.

KATSIS, C. et al. A novel method for automated {EMG} decomposition and {MUAP} classification. *Artificial Intelligence in Medicine*, v. 37, n. 1, p. 55 – 64, 2006. ISSN 0933-3657. Intelligent Data Analysis in Medicine. Disponível em: <<http://www.sciencedirect.com/science/article/pii/S0933365705001065>>. Citado 2 vezes nas páginas 14 e 20.

KIM, K. jae; HAN, I. Genetic algorithms approach to feature discretization in artificial neural networks for the prediction of stock price index. *Expert Systems with Applications*, v. 19, n. 2, p. 125 – 132, 2000. ISSN 0957-4174. Disponível em: <<http://www.sciencedirect.com/science/article/pii/S0957417400000270>>. Citado na página 24.

KLINKE, J. C.; LUCA, C. J. D. Error reduction in emg signal decomposition. *Journal of Neurophysiology*, American Physiological Society, 2014. ISSN 0022-3077. Citado na página 17.

- KUBAT, M.; HOLTE, R.; MATWIN, S. Machine learning for the detection of oil spills in satellite radar images. *Machine Learning*, Kluwer Academic Publishers, v. 30, n. 2-3, p. 195–215, 1998. ISSN 0885-6125. Disponível em: <<http://dx.doi.org/10.1023/A%3A1007452223027>>. Citado na página 36.
- LOPES, I. F. *Caracterização dos Sinais Mioelétricos dos Movimentos do Segmento Mão-Braço Através de Regressão Logística*. 2014. Trabalho de Diplomação, Curso de Engenharia de Computação, UFRGS. Citado na página 17.
- LUCA, C. J. D. et al. Decomposition of surface EMG signals. *J. Neurophysiol.*, v. 96, n. 3, p. 1646–1657, Sep 2006. Citado na página 16.
- MØLLER, M. F. A scaled conjugate gradient algorithm for fast supervised learning. *Neural Networks*, v. 6, n. 4, p. 525 – 533, 1993. ISSN 0893-6080. Disponível em: <<http://www.sciencedirect.com/science/article/pii/S0893608005800565>>. Citado na página 24.
- PATTICHIS, C.; SCHIZAS, C.; MIDDLETON, L. Neural network models in emg diagnosis. *Biomedical Engineering, IEEE Transactions on*, v. 42, n. 5, p. 486–496, May 1995. ISSN 0018-9294. Citado 4 vezes nas páginas 14, 15, 18 e 22.
- SCHONS, L. *Caracterização dos Sinais Mioelétricos dos Movimentos do Segmento Mão-Braço Através de Redes Neurais Artificiais e Algoritmos Genéticos*. 2014. Trabalho de Diplomação, Curso de Engenharia de Computação, UFRGS. Citado 2 vezes nas páginas 17 e 34.
- SUBASI, A.; YILMAZ, M.; OZCALIK, H. R. Classification of {EMG} signals using wavelet neural network. *Journal of Neuroscience Methods*, v. 156, n. 1–2, p. 360 – 367, 2006. ISSN 0165-0270. Disponível em: <<http://www.sciencedirect.com/science/article/pii/S0165027006001440>>. Citado na página 23.
- TANIKIĆ, V. D. D. Metallurgy - advances in materials and processes. 2012. Disponível em: <<http://www.intechopen.com/books/export/citation/BibTex/metallurgy-advances-in-materials-and-processes/artificial-intelligence-techniques-for-modelling-of-temperature-in-the-metal-cutting-process>>. Citado na página 23.
- TRAN, T. N.; DRAB, K.; DASZYKOWSKI, M. Revised {DBSCAN} algorithm to cluster data with dense adjacent clusters. *Chemometrics and Intelligent Laboratory Systems*, v. 120, p. 92 – 96, 2013. ISSN 0169-7439. Disponível em: <<http://www.sciencedirect.com/science/article/pii/S0169743912002249>>. Citado na página 33.
- YEGNANARAYANA, B. *ARTIFICIAL NEURAL NETWORKS*. PHI Learning, 2009. ISBN 9788120312531. Disponível em: <https://books.google.com.br/books?id=RTtvUVU_xL4C>. Citado na página 23.

Apêndices

APÊNDICE A – Função em MATLAB para MTD1

```

function [x_seg, finalCenterLocs] = seg_mtd1(x, l, q, r_target, T_lim)
%   MTD1 - metodo iterativo utilizando thresholding para deteccao de
%   centros de segmentos de comprimento constante
%
% Argumentos: (para mais detalhes, refira a descricao do MTD1)
%   x - matriz cujas colunas sao canais do sinal a ser segmentado
%   l - comprimento desejado para os segmentos
%   q - razao de atualizacao entre iteracoes para valor de threshold
%   r_target - razao minima esperada entre numero de segmentos e comprimento
%           total de sinal
%   T_lim - fracao do maximo do sinal para limite inferior de threshold
%
% Retorno:
%   x_seg - cell array com canais segmentados
%   finalCenterLocs - posicoes centrais dos segmentos

%% Preprocessamento

[L, numberOfChannels] = size(x); % comprimento do sinal e numero de canais
x_ret = abs(x); % retificacao
x_norm = zeros(L, numberOfChannels); % normalizacao
for currentChannel = 1:numberOfChannels
    x_norm(:,currentChannel) = ...
        x_ret(:,currentChannel) ./ max(x_ret(:,currentChannel));
end

%% Metodo

centerLocsCell = cell(1,numberOfChannels);
for currentChannel = 1:numberOfChannels
    T_k = 1; % canais normalizados, seus valores maximos sempre sao 1
    targetReached = false;
    while ~targetReached % processo iterativo
        T_k = q*T_k; % calcula threshold desta iteracao
        if T_k < T_lim % verifica se o limite de valor de threshold foi atingido
            break
        end
        % Identifica candidatos a centros de segmentos
        [~, centerLocsCell{1,currentChannel}] = ...

```

```

        findpeaks(double(x_norm(:,currentChannel)), ...
        'MinPeakHeight', T_k, 'MinPeakDistance',1);
    % Determina o encerramento do processo iterativo
    targetReached = (length(centerLocsCell{1,currentChannel})/L >= r_target);
end
end

%% Clustering

centerLocsArray = sort(cell2mat(centerLocsCell'));
[~, labscore] = dbscan(centerLocsArray,2000,3);
numberOfSegments = max(labscore);
finalCenterLocs = zeros(numberOfSegments,1); % medias internas aos clusters
for currentCluster = 1:numberOfSegments
    finalCenterLocs(currentCluster) = ...
        round(mean(centerLocsArray(labscore == currentCluster)));
end

%% Segmentacao

x_seg = cell(numberOfSegments, numberOfChannels);
for currentChannel = 1:numberOfChannels
    for currentSegment = 1:numberOfSegments
        switch mod(1,2)
            case 0 % se l for par
                if(finalCenterLocs(currentSegment)-1/2)<1
                    % segmento muito a esquerda
                    x_seg{currentSegment,currentChannel} = ...
                        x(1:finalCenterLocs(currentSegment)+(1/2)-1, ...
                        currentChannel);
                else if(finalCenterLocs(currentSegment)+(1/2)-1)>L
                    % segmento muito a direita
                    x_seg{currentSegment,currentChannel} = ...
                        x(finalCenterLocs(currentSegment)-1/2:L, ...
                        currentChannel);
                else
                    x_seg{currentSegment,currentChannel} = ...
                        x(finalCenterLocs(currentSegment)-1/2: ...
                        finalCenterLocs(currentSegment)+(1/2)-1, ...
                        currentChannel);
                end
            end
            case 1 % se l for impar
                if(finalCenterLocs(currentSegment) - (1-1)/2)<1
                    % segmento muito a esquerda
                    x_seg{currentSegment,currentChannel} = ...
                        x(1:finalCenterLocs(currentSegment) + (1-1)/2, ...

```

```
        currentChannel);  
    else if (finalCenterLocs(currentSegment) + (l-1)/2) > L  
        % segmento muito a direita  
        x_seg{currentSegment,currentChannel} = ...  
            x(finalCenterLocs(currentSegment) - (l-1)/2:L, ...  
              currentChannel);  
    else  
        x_seg{currentSegment,currentChannel} = ...  
            x(finalCenterLocs(currentSegment) - (l-1)/2: ...  
              finalCenterLocs(currentSegment) + (l-1)/2, ...  
              currentChannel);  
    end  
end  
end  
end  
end  
end  
end
```


APÊNDICE B – Função em MATLAB para MTD2

```
function [x_seg, finalCenterLocs] = seg_mtd2(x, l, A, B, C)
%   MTD2 - metodo nao iterativo utilizando thresholding para deteccao de
%   centros de segmentos de comprimento constante
%
% Argumentos: (para mais detalhes, refira a descricao do MTD2)
%   x - matriz cujas colunas sao canais do sinal a ser segmentado
%   l - comprimento desejado para os segmentos
%   A - coeficiente utilizado para decisao de metodo de calculo de threshold
%   B - multiplo da media aritmetica do sinal x para obtencao de threshold
%   C - fracao do valor maximo do sinal x para calculo de threshold
%
% Retorno:
%   x_seg - cell array com os canais segmentados
%   finalCenterLocs - posicoes centrais dos segmentos

%% Preprocessamento

[L, numberOfChannels] = size(x); % comprimento do sinal e numero de canais
x_ret = abs(x); % retificacao
x_norm = zeros(L, numberOfChannels); % normalizacao
for currentChannel = 1:numberOfChannels
    x_norm(:,currentChannel) = ...
        x_ret(:,currentChannel) ./ max(x_ret(:,currentChannel));
end

%% Metodo

centerLocsCell = cell(1,numberOfChannels);
for currentChannel = 1:numberOfChannels
    % Calculo do threshold
    maxValue = 1; % sinais normalizados
    meanValue = mean(x_norm(:,currentChannel));
    if maxValue > (A*meanValue)
        T = B*meanValue;
    else
        T = maxValue/C;
    end
    % Identifica centros de segmentos
    [~, centerLocsCell{1,currentChannel}] = ...
```

```

        findpeaks(double(x_norm(:,currentChannel)), ...
        'MinPeakHeight', T, 'MinPeakDistance',1);
end

%% Clustering

centerLocsArray = sort(cell2mat(centerLocsCell'));
[~, labscore] = dbscan(centerLocsArray,2000,3);
numberOfSegments = max(labscore);
finalCenterLocs = zeros(numberOfSegments,1); % medias internas aos clusters
for currentCluster = 1:numberOfSegments
    finalCenterLocs(currentCluster) = ...
        round(mean(centerLocsArray(labscore == currentCluster)));
end

%% Segmentacao

x_seg = cell(numberOfSegments, numberOfChannels);
for currentChannel = 1:numberOfChannels
    for currentSegment = 1:numberOfSegments
        switch mod(1,2)
            case 0 % se l for par
                if(finalCenterLocs(currentSegment)-1/2)<1
                    % segmento muito a esquerda
                    x_seg{currentSegment,currentChannel} = ...
                        x(1:finalCenterLocs(currentSegment)+(1/2)-1, ...
                        currentChannel);
                else if(finalCenterLocs(currentSegment)+(1/2)-1)>L
                    % segmento muito a direita
                    x_seg{currentSegment,currentChannel} = ...
                        x(finalCenterLocs(currentSegment)-1/2:L, ...
                        currentChannel);
                else
                    x_seg{currentSegment,currentChannel} = ...
                        x(finalCenterLocs(currentSegment)-1/2: ...
                        finalCenterLocs(currentSegment)+(1/2)-1, ...
                        currentChannel);
                end
            end
            case 1 % se l for impar
                if(finalCenterLocs(currentSegment) - (1-1)/2)<1
                    % segmento muito a esquerda
                    x_seg{currentSegment,currentChannel} = ...
                        x(1:finalCenterLocs(currentSegment) + (1-1)/2, ...
                        currentChannel);
                else if(finalCenterLocs(currentSegment) + (1-1)/2)>L
                    % segmento muito a direita

```

```
x_seg{currentSegment,currentChannel} = ...  
    x(finalCenterLocs(currentSegment) - (l-1)/2:L, ...  
      currentChannel);  
else  
    x_seg{currentSegment,currentChannel} = ...  
    x(finalCenterLocs(currentSegment) - (l-1)/2: ...  
      finalCenterLocs(currentSegment) + (l-1)/2, ...  
      currentChannel);  
end  
end  
end  
end  
end  
end
```

APÊNDICE C – Função em MATLAB para MTD3

```
function [x_seg, finalCenterLocs] = seg_mtd3(x, l_min, l_max, step, W, B, C)
%   MTD3 - metodo com janela deslizante para deteccao de BEP e EEP de segmentos
%   utilizando variacao total
%
% Argumentos: (para mais detalhes, refira a descricao do MTD3)
%   x - matriz cujas colunas sao canais do sinal a ser segmentado
%   l_min - comprimento minimo para segmentos
%   l_max - comprimento maximo para segmentos
%   step - numero de amostras a incrementr posicao de janela
%   W - comprimento da janela deslizante utilizada pelo metodo
%   B - valor limite para variacao total que determina um BEP
%   C - valor limite para variacao total que determina um EEP
%
% Retorno:
%   x_seg - cell array com os canais segmentados
%   finalCenterLocs - posicoes centrais dos segmentos

%% Preprocessamento
[L, numberOfChannels] = size(x); % comprimento do sinal e numero de canais
x_ret = abs(x); % retificacao
x_norm = zeros(L, numberOfChannels); % normalizacao
for currentChannel = 1:numberOfChannels
    x_norm(:,currentChannel) = ...
        x_ret(:,currentChannel) ./ max(x_ret(:,currentChannel));
end

%% Metodo

totalVariation = zeros(L-W, numberOfChannels);
BEPsLocsFlags = false(W, numberOfChannels);
EEPsLocsFlags = false(W, numberOfChannels);
BEPsLocsCell = cell(1, numberOfChannels);
EEPsLocsCell = cell(1, numberOfChannels);
searchBEP = true(numberOfChannels, 1);
lastBEPloc = zeros(numberOfChannels, 1);
for w0 = 1:step:L-W % janela deslizante para calculo de variacao total
    for currentChannel = 1:numberOfChannels
        totalVariation(w0, currentChannel) = ...
            sum(diff(x_norm(w0:w0+W, currentChannel)));
```

```

% Identificacao de BEPs e EEPs
switch searchBEP(currentChannel)
    case true % deteccao de BEPs
        if totalVariation(w0, currentChannel) > B
            BEPsLocsFlags(w0, currentChannel) = true;
            lastBEPloc(currentChannel) = w0;
            searchBEP(currentChannel) = false;
        end
    case false % deteccao de EEPs
        if (w0+W-lastBEPloc(currentChannel)) > l_max
            % segmento excederia comprimento maximo
            BEPsLocsFlags(lastBEPloc(currentChannel),currentChannel) = false;
            searchBEP(currentChannel) = true;
        else if (totalVariation(w0, currentChannel) < C) && ...
            (w0+W-lastBEPloc(currentChannel) > l_min)
            EEPsLocsFlags(w0+W,currentChannel) = true;
            searchBEP(currentChannel) = true;
        end
    end
end

end
end
end
for currentChannel = 1:numberOfChannels
    BEPsLocsCell{1,currentChannel} = find(BEPsLocsFlags(:,currentChannel));
    EEPsLocsCell{1,currentChannel} = find(EEPsLocsFlags(:,currentChannel));
end

%% Clustering

BEPsLocsArray = sort(cell2mat(BEPsLocsCell'));
EEPsLocsArray = sort(cell2mat(EEPsLocsCell'));
[~, labscoreBEPs] = dbscan(BEPsLocsArray,2000,3);
[~, labscoreEEPs] = dbscan(EEPsLocsArray,2000,3);
numberOfBEPs = max(labscoreBEPs);
numberOfEEPs = max(labscoreEEPs);
% medias internas aos clusters
meanBEPs = zeros(numberOfBEPs,1);
for currentCluster = 1:numberOfBEPs
    meanBEPs(currentCluster) = ...
        round(mean(BEPsLocsArray(labscoreBEPs == currentCluster)));
end
meanEEPs = zeros(numberOfEEPs,1);
for currentCluster = 1:numberOfEEPs
    meanEEPs(currentCluster) = ...
        round(mean(EEPsLocsArray(labscoreEEPs == currentCluster)));
end

```

```

%% Pareamento final de BEPs e EEPs
% (devem ocorrer alternadamente e atender requisitos de l_min e l_max)

allLocs = sortrows([meanBEPs,true(length(meanBEPs),1);meanEEPs,false(length(meanEEPs),1)]);
locsDelta = diff(allLocs,1);
numberOfDeltas = length(locsDelta(:,1));
finalBEPsFlags = false(numberOfDeltas+1,1);
finalEEPsFlags = false(numberOfDeltas+1,1);
for currentDelta = 1:numberOfDeltas
    if (locsDelta(currentDelta,1) > l_min) && ...
        (locsDelta(currentDelta,1) < l_max) && ...
        locsDelta(currentDelta,2) == -1; % par BEP-EEP valido
        finalBEPsFlags(currentDelta) = true;
        finalEEPsFlags(currentDelta+1) = true;
    end
end
finalBEPs = allLocs(finalBEPsFlags,1);
finalEEPs = allLocs(finalEEPsFlags, 1);

%% Segmentacao

numberOfSegments = length(finalBEPs);
finalCenterLocs = zeros(numberOfSegments,1);
x_seg = cell(numberOfSegments,numberOfChannels);
for currentChannel = 1:numberOfChannels
    for currentSegment = 1:numberOfSegments
        x_seg{currentSegment,currentChannel} = ...
            x(finalBEPs(currentSegment):finalEEPs(currentSegment),currentChannel);
        finalCenterLocs(currentSegment) = ...
            round(mean([finalBEPs(currentSegment),finalEEPs(currentSegment)]));
    end
end
end
end

```

APÊNDICE D – Função em MATLAB para MTD4

APÊNDICE E – Código Utilizado para Medida de Número de Segmentos Obtidos para Diferentes Parâmetros com MTD1

```
% Implementacao do MTD1 com variacao de parametros
% e medida do numero de segmentos obtidos por movimento
close all
clear

%% Parametros utilizados
% Predeterminados
l = 10e3;
r_target = 5.6e-5;

%% Base de dados Ninapro

ninaproList = ls('database/ninapro2/S*_E1*');
numberOfSubjects = length(ninaproList);
numberOfClasses = 17;

%% Teste das diferentes combinacoes
numberOfSegments = zeros(numberOfCombinations,numberOfSubjects,numberOfClasses);
parfor_progress(numberOfSubjects);
parfor currentSubject = 1:numberOfSubjects
    S = load(['database/ninapro2/' ninaproList(currentSubject,:)]);
    % Combinacoes a serem testadas
    q = [0.8 0.85 0.9 0.95];
    T_lim = [0.05 0.1 0.15 0.2];
    combinations = combvec(q, T_lim)';
    numberOfCombinations = length(combinations);
    for currentCombination = 1:numberOfCombinations
        % metodo de segmentacao
        [x_seg, centerLocs] = ...
            seg_mtd1(S.emg, l, combinations(currentCombination,1), r_target, ...
                combinations(currentCombination,2));
        % identificacao do movimento correspondente a cada segmento
        targetClasses = identifyClasses(centerLocs, S.stimulus);
        % numero de segmentos obtidos por movimento
        for currentClass = 1:numberOfClasses
            numberOfSegments(currentCombination,currentSubject,currentClass) = ...
```



```
        length(find(targetClasses(:,currentClass)));  
    end  
end  
S = []; % libera espaco da memoria  
parfor_progress; % exhibe progresso do parfor  
end  
meanNumberOfSegments = squeeze(mean(numberOfSegments,2));  
save('./out/workspace/numbers/number_MTD1.mat') % salva a workspace atual
```

APÊNDICE F – Código Utilizado para Medida de Número de Segmentos Obtidos para Diferentes Parâmetros com MTD2

```
% Implementacao do MTD2 com variacao de parametros
% e medida do numero de segmentos obtidos por movimento
close all
clear

%% Parametros utilizados
l = 10e3;

%% Base de dados Ninapro

ninaproList = ls('database/ninapro2/S*_E1*');
numberOfSubjects = length(ninaproList);
numberOfClasses = 17;

%% Teste das diferentes combinacoes
numberOfSegments = zeros(numberOfCombinations,numberOfSubjects,numberOfClasses);
parfor_progress(numberOfSubjects);
parfor currentSubject = 1:numberOfSubjects
    S = load(['database/ninapro2/' ninaproList(currentSubject,:)]);
    % Combinacoes a serem testadas
    A = [20 30 40];
    B = [2 5 8];
    C = [2 5 8];
    combinations = combvec(A, B, C)';
    numberOfCombinations = length(combinations);
    for currentCombination = 1:numberOfCombinations
        % metodo de segmentacao
        [x_seg, centerLocs] = ...
            seg_mtd2(S.emg, l, combinations(currentCombination,1), ...
                combinations(currentCombination,2), combinations(currentCombination,3));
        % identificacao do movimento correspondente a cada segmento
        targetClasses = identifyClasses(centerLocs, S.stimulus);
        % numero de segmentos obtidos por movimento
        for currentClass = 1:numberOfClasses
            numberOfSegments(currentCombination,currentSubject,currentClass) = ...
                length(find(targetClasses(:,currentClass)));
        end
    end
end
```

```
        end
    end
    S = []; % libera espaco da memoria
    parfor_progress; % exhibe progresso do parfor
end
meanNumberOfSegments = squeeze(mean(numberOfSegments,2));
save('./out/workspace/numbers/number_MTD2.mat') % salva a workspace atual
```

APÊNDICE G – Código Utilizado para Medida de Número de Segmentos Obtidos para Diferentes Parâmetros com MTD3

```
% Implementacao do MTD3 com variacao de parametros
% e medida do numero de segmentos obtidos por movimento
close all
clear

%% Parametros utilizados
l_min = 7.5e3;
l_max = 12.5e3;
step = 100;
W = 5e3;

%% Base de dados Ninapro

ninaproList = ls('database/ninapro2/S*_E1*');
numberOfSubjects = length(ninaproList);
numberOfClasses = 17;

%% Teste das diferentes combinacoes
numberOfSegments = zeros(numberOfCombinations,numberOfSubjects,numberOfClasses);
parfor_progress(numberOfSubjects);
parfor currentSubject = 1:numberOfSubjects
    S = load(['database/ninapro2/' ninaproList(currentSubject,:)]);
    % Combinacoes a serem testadas
    B = [0.5 0.1 0.15 0.2];
    C = [-0.5 -0.1 -0.15 -0.2];
    combinations = combvec(B, C)';
    numberOfCombinations = length(combinations);
    for currentCombination = 1:numberOfCombinations
        % metodo de segmentacao
        [x_seg, centerLocs] = ...
            seg_mtd3(S.emg, l_min, l_max, step, W, ...
                combinations(currentCombination,1), combinations(currentCombination,2));
        % identificacao do movimento correspondente a cada segmento
        targetClasses = identifyClasses(centerLocs, S.stimulus);
        % numero de segmentos obtidos por movimento
        for currentClass = 1:numberOfClasses
```

```
        numberOfSegments(currentCombination,currentSubject,currentClass) = ...  
            length(find(targetClasses(:,currentClass)));  
    end  
end  
S = []; % libera espaco da memoria  
parfor_progress; % exhibe progresso do parfor  
end  
meanNumberOfSegments = squeeze(mean(numberOfSegments,2));  
save('./out/workspace/numbers/number_MTD3.mat') % salva a workspace atual
```

APÊNDICE H – Código Utilizado para Medida de Número de Segmentos Obtidos para Diferentes Parâmetros com MTD4

APÊNDICE I – Código Utilizado para Obtenção de Resultados de Classificação Utilizando RNA com MTD1

```
% Implementacao do MTD1 com parametros fixos
% e resultados de classificacao utilizando RNA
close all
clear

%% Parametros utilizados

l = 10e3;
r_target = 5.6e-5;
q = 0.95;
T_lim = 0.1;

%% Base de dados Ninapro

ninaproList = ls('database/ninapro2/S*_E1*');
numberOfSubjects = length(ninaproList);
numberOfChannels = 12;

%% Implementacao com treinamento interno a cada voluntario

numberOfFeatures = 3;
numberOfMoves = 17;
predictorsCellArray = cell(1, numberOfSubjects);
targetsCellArray = cell(1, numberOfSubjects);
internalClassificationCellArray = cell(1, numberOfSubjects);
centerLocsCellArray = cell(1, numberOfSubjects);
numberOfSegPerMove = zeros(numberOfSubjects, numberOfMoves);
trainingRecords = struct();
for currentSubject = 1:numberOfSubjects
    fprintf('currentSubject = %i / %i\n', currentSubject, numberOfSubjects)
    % Carrega o voluntario atual
    load(['database/ninapro2/' ninaproList(currentSubject,:)])

    % Segmentacao
    [x_seg, centerLocsCellArray{1, currentSubject}] = ...
        seg_mtd1(emg, l, q, r_target, T_lim);
```

```
% Alvos para treinamento da rede neural
targetsCellArray{1, currentSubject} = ...
    identifyClasses(centerLocsCellArray{1,currentSubject},stimulus);

% Numero de segmentos por movimento
numberOfSegPerMove(currentSubject,:) = sum(targetsCellArray{1, currentSubject});
% Divisao de dados para treinamento
numberOfSegments = length(centerLocsCellArray{1,currentSubject});
trainIndFlags = false(numberOfSegments,1);
valIndFlags = false(numberOfSegments,1);
testIndFlags = false(numberOfSegments,1);
for currentMove = 1:numberOfMoves
    counter = 0;
    for currentSegment = 1:numberOfSegments
        if targetsCellArray{1,currentSubject}(currentSegment,currentMove)
            counter = counter + 1;
            if counter < 5
                trainIndFlags(currentSegment) = true;
            else if counter == 5
                valIndFlags(currentSegment) = true;
            else if counter > 5
                testIndFlags(currentSegment) = true;
            end
        end
    end
end
end
trainInd = find(trainIndFlags);
valInd = find(valIndFlags);
testInd = find(testIndFlags);

% Preditores da RNA
predictorsCellArray{1, currentSubject} = ...
    zeros(numberOfSegments, numberOfFeatures*numberOfChannels);
for currentSegment = 1:numberOfSegments
    for currentChannel = 1:numberOfChannels
        % RMS
        predictorsCellArray{1, currentSubject}...
            (currentSegment, currentChannel + 0*numberOfChannels) = ...
            rms(x_seg{currentSegment, currentChannel});
        % Variancia
        predictorsCellArray{1, currentSubject} ...
            (currentSegment, currentChannel + 1*numberOfChannels) = ...
            var(x_seg{currentSegment, currentChannel});
        % Frequencia mediana
```



```

        predictorsCellArray{1, currentSubject} ...
            (currentSegment, currentChannel + 2*numberOfChannels) = ...
            medfreq(x_seg{currentSegment, currentChannel});
    end
end

% Treinamento de rede neural
net = patternnet(10, 'trainscg');
net.divideFcn = 'divideind';
net.divideParam.trainInd = trainInd;
net.divideParam.valInd = valInd;
net.divideParam.testInd = testInd;
[internalTrainedNet, trainingRecords(currentSubject).tr] = train(net, ...
    predictorsCellArray{1, currentSubject}', ...
    targetsCellArray{1, currentSubject}');
% Classificacao
internalClassificationCellArray{1, currentSubject} = ...
    internalTrainedNet(predictorsCellArray{1, currentSubject}');

% Plot da matriz de confusao
plotconfusion(targetsCellArray{1, currentSubject}', ...
    internalClassificationCellArray{1, currentSubject}, ...
    ['Current Subject: ' num2str(currentSubject) ...
    ' Segmentation Method: MTD1'])
savefig(['./out/confusion/S' num2str(currentSubject) '_MTD1.fig'])
end
save('./out/workspace/rna_MTD1.mat') % salva a workspace atual

```

APÊNDICE J – Código Utilizado para Obtenção de Resultados de Classificação Utilizando RNA com MTD2

```
% Implementacao do MTD2 com parametros fixos
% e resultados de classificacao utilizando RNA
close all
clear

%% Parametros utilizados

l = 10e3;
A = 30;
B = 8;
C = 2;

%% Base de dados Ninapro

ninaproList = ls('database/ninapro2/S*_E1*');
numberOfSubjects = length(ninaproList);
numberOfChannels = 12;

%% Implementacao com treinamento interno a cada voluntario

numberOfFeatures = 3;
numberOfMoves = 17;
predictorsCellArray = cell(1, numberOfSubjects);
targetsCellArray = cell(1, numberOfSubjects);
internalClassificationCellArray = cell(1, numberOfSubjects);
centerLocsCellArray = cell(1, numberOfSubjects);
numberOfSegPerMove = zeros(numberOfSubjects,numberOfMoves);
trainingRecords = struct();
for currentSubject = 1:numberOfSubjects
    fprintf('currentSubject = %i / %i\n', currentSubject, numberOfSubjects)
    % Carrega o voluntario atual
    load (['database/ninapro2/' ninaproList(currentSubject,:)])

    % Segmentacao
    [x_seg, centerLocsCellArray{1, currentSubject}] = ...
        seg_mtd2(emg, l, A, B,C);
```

```
% Alvos para treinamento da rede neural
targetsCellArray{1, currentSubject} = ...
    identifyClasses(centerLocsCellArray{1,currentSubject},stimulus);

% Numero de segmentos por movimento
numberOfSegPerMove(currentSubject,:) = sum(targetsCellArray{1, currentSubject});
% Divisao de dados para treinamento
numberOfSegments = length(centerLocsCellArray{1,currentSubject});
trainIndFlags = false(numberOfSegments,1);
valIndFlags = false(numberOfSegments,1);
testIndFlags = false(numberOfSegments,1);
for currentMove = 1:numberOfMoves
    counter = 0;
    for currentSegment = 1:numberOfSegments
        if targetsCellArray{1,currentSubject}(currentSegment,currentMove)
            counter = counter + 1;
            if counter < 5
                trainIndFlags(currentSegment) = true;
            else if counter == 5
                valIndFlags(currentSegment) = true;
            else if counter > 5
                testIndFlags(currentSegment) = true;
            end
        end
    end
end
end
trainInd = find(trainIndFlags);
valInd = find(valIndFlags);
testInd = find(testIndFlags);

% Preditores da RNA
predictorsCellArray{1, currentSubject} = ...
    zeros(numberOfSegments, numberOfFeatures*numberOfChannels);
for currentSegment = 1:numberOfSegments
    for currentChannel = 1:numberOfChannels
        % RMS
        predictorsCellArray{1, currentSubject}...
            (currentSegment, currentChannel + 0*numberOfChannels) = ...
            rms(x_seg{currentSegment, currentChannel});
        % Variancia
        predictorsCellArray{1, currentSubject} ...
            (currentSegment, currentChannel + 1*numberOfChannels) = ...
            var(x_seg{currentSegment, currentChannel});
        % Frequencia mediana
```

```

        predictorsCellArray{1, currentSubject} ...
            (currentSegment, currentChannel + 2*numberOfChannels) = ...
            medfreq(x_seg{currentSegment, currentChannel});
    end
end

% Treinamento de rede neural
net = patternnet(10, 'trainscg');
net.divideFcn = 'divideind';
net.divideParam.trainInd = trainInd;
net.divideParam.valInd = valInd;
net.divideParam.testInd = testInd;
[internalTrainedNet, trainingRecords(currentSubject).tr] = train(net, ...
    predictorsCellArray{1, currentSubject}', ...
    targetsCellArray{1, currentSubject}');
% Classificacao
internalClassificationCellArray{1, currentSubject} = ...
    internalTrainedNet(predictorsCellArray{1, currentSubject}');

% Plot da matriz de confusao
plotconfusion(targetsCellArray{1, currentSubject}', ...
    internalClassificationCellArray{1, currentSubject}, ...
    ['Current Subject: ' num2str(currentSubject) ...
    ' Segmentation Method: MTD2'])
savefig(['./out/confusion/S' num2str(currentSubject) '_MTD2.fig'])
end
save('./out/workspace/rna_MTD2.mat') % salva a workspace atual

```

APÊNDICE K – Código Utilizado para Obtenção de Resultados de Classificação Utilizando RNA com MTD3

```
% Implementacao do MTD3 com parametros fixos
% e resultados de classificacao utilizando RNA
close all
clear

%% Parametros utilizados

l_min = 7500;
l_max = 12500;
W=5000;
step = 500;
B = 0.04;
C = -0.01;

%% Base de dados Ninapro

ninaproList = ls('database/ninapro2/S*_E1*');
numberOfSubjects = length(ninaproList);
numberOfChannels = 12;

%% Implementacao com treinamento interno a cada voluntario

numberOfFeatures = 3;
numberOfMoves = 17;
predictorsCellArray = cell(1, numberOfSubjects);
targetsCellArray = cell(1, numberOfSubjects);
internalClassificationCellArray = cell(1, numberOfSubjects);
centerLocsCellArray = cell(1, numberOfSubjects);
numberOfSegPerMove = zeros(numberOfSubjects,numberOfMoves);
trainingRecords = struct();
for currentSubject = 1:numberOfSubjects
    fprintf('currentSubject = %i / %i\n', currentSubject, numberOfSubjects)
    % Carrega o voluntario atual
    load (['database/ninapro2/' ninaproList(currentSubject,:)])

    % Segmentacao
```

```
[x_seg, centerLocsCellArray{1, currentSubject}] = ...
    seg_mtd3(emg, l_min, l_max, step, W, B, C);

% Alvos para treinamento da rede neural
targetsCellArray{1, currentSubject} = ...
    identifyClasses(centerLocsCellArray{1, currentSubject}, stimulus);

% Numero de segmentos por movimento
numberOfSegPerMove(currentSubject, :) = sum(targetsCellArray{1, currentSubject});
% Divisao de dados para treinamento
numberOfSegments = length(centerLocsCellArray{1, currentSubject});
trainIndFlags = false(numberOfSegments, 1);
valIndFlags = false(numberOfSegments, 1);
testIndFlags = false(numberOfSegments, 1);
for currentMove = 1:numberOfMoves
    counter = 0;
    for currentSegment = 1:numberOfSegments
        if targetsCellArray{1, currentSubject}(currentSegment, currentMove)
            counter = counter + 1;
            if counter < 5
                trainIndFlags(currentSegment) = true;
            else if counter == 5
                valIndFlags(currentSegment) = true;
            else if counter > 5
                testIndFlags(currentSegment) = true;
            end
        end
    end
end
trainInd = find(trainIndFlags);
valInd = find(valIndFlags);
testInd = find(testIndFlags);

% Preditores da RNA
predictorsCellArray{1, currentSubject} = ...
    zeros(numberOfSegments, numberOfFeatures*numberOfChannels);
for currentSegment = 1:numberOfSegments
    for currentChannel = 1:numberOfChannels
        % RMS
        predictorsCellArray{1, currentSubject}...
            (currentSegment, currentChannel + 0*numberOfChannels) = ...
            rms(x_seg{currentSegment, currentChannel});
        % Variancia
        predictorsCellArray{1, currentSubject} ...
            (currentSegment, currentChannel + 1*numberOfChannels) = ...
```

```

        var(x_seg{currentSegment, currentChannel});
    % Frequencia mediana
    predictorsCellArray{1, currentSubject} ...
        (currentSegment, currentChannel + 2*numberOfChannels) = ...
        medfreq(x_seg{currentSegment, currentChannel});

    end
end

% Treinamento de rede neural
net = patternnet(10, 'trainscg');
net.divideFcn = 'divideind';
net.divideParam.trainInd = trainInd;
net.divideParam.valInd = valInd;
net.divideParam.testInd = testInd;
[internalTrainedNet, trainingRecords(currentSubject).tr] = train(net, ...
    predictorsCellArray{1, currentSubject}', ...
    targetsCellArray{1, currentSubject}');
% Classificacao
internalClassificationCellArray{1, currentSubject} = ...
    internalTrainedNet(predictorsCellArray{1, currentSubject}');

% Plot da matriz de confusao
fig = figure()
plotconfusion(targetsCellArray{1, currentSubject}', ...
    internalClassificationCellArray{1, currentSubject}, ...
    ['Current Subject: ' num2str(currentSubject) ...
    ' Segmentation Method: MTD3'])
savefig(['./out/confusion/S' num2str(currentSubject) '_MTD3.fig'])
set (fig, 'Units', 'normalized', 'Position', [0,0,1,1]);
end
save('./out/workspace/rna_MTD3.mat') % salva a workspace atual

```

APÊNDICE L – Código Utilizado para Obtenção de Resultados de Classificação Utilizando RNA com MTD4

APÊNDICE M – Código utilizado para determinação de r_{target} do MTD1

```
% Determinacao de r_target para MTD1

%% Base de dados Ninapro

ninaproList = ls('database/ninapro2/S*_E1*');
numberOfSubjects = length(ninaproList);
numberOfSegments = 102; % 17 movimentos * 6 repeticoes

%% Obtencao de r_target da database

r_target = zeros(numberOfSubjects, 1);

for currentSubject = 1:numberOfSubjects

    fprintf('currentSubject = %i\n',currentSubject)

    % Carrega o voluntario atual
    load(['database/ninapro2/' ninaproList(currentSubject,:)])

    % r_target para este voluntario
    r_target(currentSubject) = numberOfSegments/length(emg);

end

min_r_target = min(r_target)
```

APÊNDICE N – Função em MATLAB para agrupamento com DBSCAN

```

function [labs labscore] = dbscan(a,Eps,MinPts)
% DBSCAN clustering
% [labs labscore] = dbscan(A,Eps,MinPts)
%
% DBSCAN clustering of data matrix in A. labels is a vector with
% cluster labels for each vector.
%
% In case of publication of any application of this method,
% please, cite the original work:
% Thanh N. Tran*, Klaudia Drab, Michal Daszykowski, "Revised DBSCAN algorithm
% to cluster data with dense adjacent clusters", Chemometrics and Intelligent
% Laboratory Systems, 120:92-96.
% DOI: 10.1016/j.chemolab.2012.11.006

UNCLASSIFIED = 0;
BORDER = -2;

% Square Eps in order not to square all distances of points
%eps = eps^2;

m = size(a,1);
labs = zeros(m,1);
ClusterId = 1;
for i=1:m
    if labs(i) == UNCLASSIFIED

        % Expand cluster ClusterId
        % Get a set of points of distance < eps
        [ExpandClusterReturn labs]= ExpandCluster(a,labs,i,ClusterId,Eps, MinPts);
        if ExpandClusterReturn
            ClusterId = ClusterId +1;
        end
    end
end

% Step 3:
labscore = labs; core_index = find(labscore > 0);
border_points = find(labs==BORDER);
% For xborder in border_list but has no ClusterId

```

```

    for i=1:length(border_points)
        % xborder in border_list
        currentB = border_points(i);
        d=distance(+a(currentB,:),+a(core_index,:),1);
        % the closest core-points
        [tmp nearest_core]=min(d);
        nearest_core_index=core_index(nearest_core);
        %Assign xborder to ClusterId of the closest core-points
        labs(currentB)=labs(nearest_core_index);
    end
end

function [ExpandClusterReturn labs]= ExpandCluster(a,labs,i,ClusterId, Eps, MinPts)
UNCLASSIFIED = 0;
NOISE = -1;
BORDER = -2;
    % calculate distances
    d=distance(+a(i,:),+a(:,,:),1);

    % seeds = Retrieve_Neighbors(xi, Eps)
    seeds = find(d < Eps);

    % If |seeds| < MinPts
    if size(seeds,2) < MinPts,
        labs(i) = NOISE; % Assign xi as noise
        ExpandClusterReturn = 0; % Return without expansion success
    else
        % STEP 1: xi is identified as a starting core-point for ClusterId
        %labs(i) = ClusterId; % Thanh changed in May 2012
        % Exclude xi from the Seeds
        %seeds = setdiff(seeds, i); % Thanh changed in May 2012
        % STEP 2: Identify chains
        % For all xj in seeds
        while ~isempty(seeds) % Not an empty seeds
            % current point is the first point in seeds

            currentP = seeds(1);
            d=distance(+a(currentP,:),+a(:,,:),1);
            % NEps(xj)= Retrieve_Neighbors(xj,Eps)
            result = find(d <= Eps);
            %If | NEps(xj) | >= MinPts
            // xj is a core point
            if length(result) >= MinPts
                % Assign xj to ClusterId
                labs(currentP) = ClusterId;
                % Add all UNCLASSIFIED in NEps(xj) to seeds
                result_unclassified = result(find(labs(result)==UNCLASSIFIED));
            end
            % Remove currentP from seeds
            seeds = setdiff(seeds, currentP);
        end
    end
    ExpandClusterReturn = labs(i);
end

```

```
        result_noise = result(find(labs(result)==NOISE));
        % Temporary complete the intermediate chain ... the chain can
        % The border points have not yet been assigned to any cluster
        labs([result_unclassified result_noise]) = BORDER;
    % first assign to a border-point... later will be reassigned to e.g. core-point
        seeds = union(seeds,result_unclassified);
    end
    % Exclude the current point in seeds and go back to the loop
    seeds = seeds(2:size(seeds,2));
end % end while
% Return with expansion success
ExpandClusterReturn = 1;    % return true
end
end
```

APÊNDICE O – Função em MATLAB para obtenção dos movimentos relacionados aos segmentos

```
function targetDataArray = identifyClasses(centerLocs, stimulus)

%% Remodela o vetor de estímulo, removendo trechos de 0

reshapedStimulus = stimulus;
currentMovement = 1;
for index = 1:length(stimulus)
    if reshapedStimulus(index) ~= 0
        currentMovement = reshapedStimulus(index);
    else
        reshapedStimulus(index) = currentMovement;
    end
end

%% Gera a matriz de identificacao das classes

numberOfSegments = length(centerLocs);
numberOfClasses = 17;
targetDataArray = false(numberOfSegments, numberOfClasses);
for currentSegment = 1:numberOfSegments
    targetDataArray(currentSegment,...
        reshapedStimulus(centerLocs(currentSegment))) = true;
end

end
```